Leveraging Reinforcement Learning for Autonomous Data Pipeline Optimization and Management

Chandrakanth Lekkala

Email: *chan.lekkala[at]gmail.com*

Abstract: ***The growing advancement of data pipelines in large - scale data processing systems presents many problems for efficient resource management and optimization. Reinforcement Learning (RL) was developed as an efficient method to handle complex systems smartly with application in data pipeline autonomy and optimization. The essay examines implementing RL techniques to optimize data pipelines and resource distribution in big data processing systems. We discuss the problems of applying RL in data pipeline situations, like defining the reward functions and the delayed feedback. We also examine the RL algorithms like Q learning and Policy Gradients. The two sections summarise findings of recent research and case studies, in which they point out how RL can help improve data pipeline competence regarding resource usage and a quick response to changes within workloads. Lastly, we explore emerging research trends and the unsolved problems in the RL - driven data pipeline optimization area.***

Keywords: Reinforcement Learning, Data Pipelines, Autonomous Optimization, Resource Management, Q - learning, Policy Gradients

1. Introduction

Data pipelines have become vital to a large - scale enterprise data processing environment, facilitating data transfer from various stages, from ingestion to processing and serving [1]. Nevertheless, the rising level of complication associated with data channels is a source of major management and optimization challenges due to the resources [2]. Traditional optimization techniques, like manual tuning and heuristics, usually yield suboptimal results and resource utilization, especially in a changing environment with different workloads [3].

Reinforcement Learning (RL) has gained pronounced recognition in recent decades as a powerful paradigm aimed at self - organizing and controlling complex systems [4]. In RL, an agent learns how to make the best decisions in interaction with an environment by maximizing a cumulative reward signal [5]. The emergence of deep reinforcement learning techniques, which have been on the rise since the mid - 2010s, has further improved the applicability and effectiveness of RL in different fields, such as robotics, gaming, and system optimization [6].

Moreover, the popularity of reinforcement learning has also been fueled by high milestone achievements such as AlphaGo beating human Go champions in 2016 [7] and OpenAI creating a Dota 2 bot in 2018 [8] that exhibited RL capabilities in complex decision situations. In this paper, we will give an application example of the heavy usage of RL techniques for tuning data pipelines and job schedules in mass data processing setups. In the data pipeline contexts, we talk about the difficulties of using RL, for example, the definition of the appropriate reward functions, the handling of delayed feedback and the exploration - exploitation trade - off. We now look closely into their application of popular reinforcement learning methods, such as Q - learning and Policy Gradient, which are the means to these ends and guide such data processing pipelines.

2. Background and Related Work

2.1 Data Pipelines and the Challenges of Optimization

Data pipelines represent an organization's actions to transform raw data into valuable information, actionable data, and knowledge. A classical data pipeline usually comprises several stages, namely, data ingestion, preprocessing, transformation, analysis, and data serving, as depicted in Fig.1.

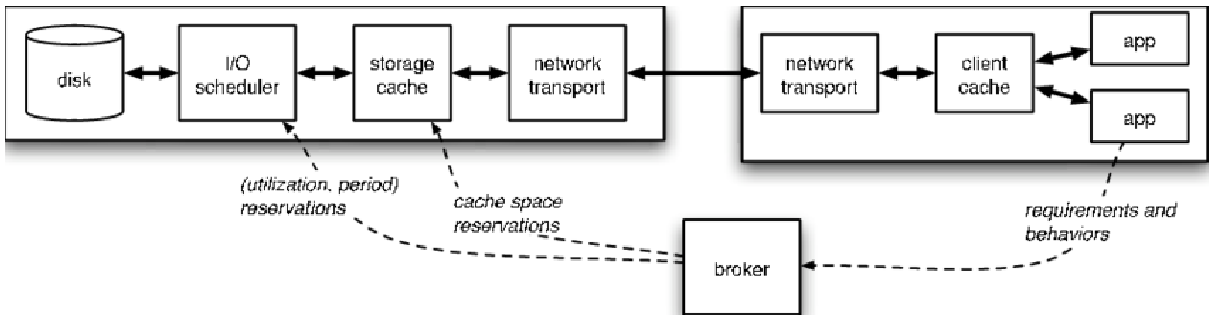

Figure 1: Typical stages in a data pipeline [10]

Efficiently defining and distributing resources across these stages is an important factor for efficient work, cutting costs, and meeting service - level objectives (SLOs) [11]. Nevertheless, data pipeline resource optimization is a challenging task. The reason is conditions such as workload dynamicity, unpredictability, complex resource dependencies, heterogeneity of infrastructure, and, ultimately, multiple optimization objectives [12].

Conventional methods of data pipeline optimization, i. e., manual tuning, heuristics, and rule - based systems, are usually unable to adequately tackle these challenges [13]. They rely on human expertise and established policies, which can be slow and subject to errors and May not be the best course of action, especially in changing and complex situations.

2.2 Reinforcement Learning for Autonomous Optimization

Reinforcement Learning (RL) is a kind of machine learning (ML) that concentrates on learning optimal decision - making policies through interactions with an environment [14]. In an RL setting, an agent receives a current state from the environment, proceeds with its action according to the policy, and then gets an award or penalty when the outcome of its action is known. The goal for the agent is to acquire a policy that actualizes the highest value possible of the expected total reward over time.
RL has several fundamental components and concepts, such as states, actions, rewards, policies, and value functions [15]. RL techniques may be subdivided into two classes of AI algorithms: value - based approaches (e. g. Q - learning and SARSA) and policy - based approaches (e. g. Policy Gradients and Actor - Critic).
Robotics - applied learning is most advantageous in this case because the capability to learn from experience and adjust to a dynamically responsive ambience is available without any models or labels. Through the state and action spaces and the rewards' feedback received, an RL agent can discover the optimal policies that will maximize long - term performance even in conditions of uncertainty and variability.

2.3 RL Uses for System Optimization Problems

Reinforcement learning has been applied in solving system optimization problems and proved the automatic decision - making system's capability in complex environments. Some notable examples include:

a) Resource Management in Cloud Computing: RL has been applied to optimize resource allocation, task scheduling and auto - scaling in cloud computing environments; it adapts to dynamic workloads and minimizes the costs [18], [19].
b) Network Traffic Engineering: RL - based models have been suggested to provide superior effectiveness in network routing, congestion control, and quality of service (QoS) in SDNs and wireless networks [20], [21].
c) Database Management: With RL implemented in autonomous database tuning, index selection, and query optimization, the learning machine would gain experience through the workload pattern and adjust to the changing data distribution [22], [23].
d) Energy Optimization: This technique has been applied in data centres, smart grids and renewable energy systems to optimize energy consumption and performance [24] [25].

These use cases will serve as proofs of concept, showing the variety and efficiency of RL's application to system optimization. This will also make RL more feasible in data pipeline optimization.

3. Reinforcement Learning was applied to Data Pipeline Optimization in the first place.

3.1 Problem Formulation

Data pipeline optimization defines the problem as a Markov Decision Process (MDP) [26] when applying reinforcement learning to the situation. An MDP is defined by a tuple (S, A, P, R), where: An MDP is defined by a tuple (S, A, P, R), where:

- S denotes the set of all states indicating either system statuses like resource utilization, progress of the task and data parameters or simply states of the pipeline itself.
- A covers the whole set of tasks that the RL agent must complete to get the best results out of the pipeline (task distribution, job to perform scheduling, and data processing).
- P is the state transition probability function, which characterizes the probability of moving from one state to another under the given action.

R represents the reward function, which evaluates state actions through a scalar value mapping and becomes the sign of this state - action pair's desirability.

The RL agent's goal is to learn a policy π: S → V that yields the highest expected win sequence in a finite or infinite timeframe.

3.2 Two challenges and considerations.

Applying RL to data pipeline optimization presents several challenges and considerations: Applying RL to data pipeline optimization presents several challenges and considerations:

a) State and Action Space Design: An equally important process is defining the ideal state space and the state the robot needs to learn to work with. State space construction should involve features that could influence the pipe performance. Similarly, the action space should contain all the optimization decisions [27]. Nevertheless, designing compact and succinct state and action spaces can be difficult, particularly for complex and dynamic pipelines.
b) Reward Function Design: The reward function is interpreted as the critical aspect of RL because it points to the path of how the agent learns and behaves. Creating the gratification function for the data pipeline optimization problem is no easy feat and takes into account factors including latency, throughput, cost, and reliability [28]. The reward function should offer proper feedback to the agent and prevent the agent from the unintended effects of gaming.
c) Delayed Feedback and Credit Assignment: It is sometimes hard to see how the effects of one's conduct on the system's performance may happen after a certain time, thus making

sense that the reward be delayed [29]. Consequently, the agent may need to identify often - specific actions that are notable for the policy. The procedures like temporal difference learning and eligibility traces can be used to overcome this obstacle [30]

d) Exploration - Exploitation Trade - off: HR agents must hurry the deployment of new actions and the accumulation of a known policy in parallel. Poor exploration will cause the formulation of inadequate measures; on the contrary, overshooting will produce deformations of poor performance and excessive wasting of resources. The ε - greedy, Upper Confidence Bound (UCB), and Thompson sampling algorithms are the techniques used to deal with this trade - off [32].
e) Scalability and Computational Complexity: Learning in Data pipelines tends to be a big - scale and higher - dimensional space case, which puts demand on computational complexity and is slow to convergent [33]. Techniques like function approximation, hierarchical RL, and parallel learning could be applied to scaling large - scale.
f) Transfer Learning and Generalization: Pipelines could be the same for different workloads and environments. Transfer learning of knowledge previously gained through one piece of the pipeline can enhance data efficiency and generalization between one RL agent and another [35]. Despite this, smooth and productive translation of knowledge in norstatic and diverse surroundings is a challenge.

Three Reinforcement Learning Algorithms for Data Pipeline Optimization

Multiple decent RL algorithms have been formulated and used for the data pipeline optimization issues, each with pros and cons. In this section, we discuss two popular algorithm families: Q - learning and policy gradients, two commonly used RL algorithms.

Q learning: The Q - learning algorithm is a value - based RL algorithm that learns the action value function Q (s, a), that is, the expected cumulative reward of taking action in state s. [36]. The Q - function is updated iteratively using the Bellman equation: The Q - function is updated iteratively using the Bellman equation:

$Q (s, a) \leftarrow Q (s, a) + \alpha [r + \gamma \max (a') Q (s', a') - Q (s, a)]$

Where α is the learning rate, γ is the discount factor, r is the immediate reward, and the next state.

Q learning has been applied to various data pipeline optimization problems, such as task scheduling [37], resource allocation [38], and data flow control [39]. Nevertheless, Q - learning is subject to overestimation bias and instability. Complexity even more comes into play in the case of high - dimensional and continuous state and action spaces [40].

Policy Gradients: Policy Gradient methods directly learn a parameterized policy π (a|s, θ) that maps state - to - action probabilities [41]. The policy parameters θ are updated using gradient ascent on the expected cumulative reward J (θ): The policy parameters θ are updated using gradient ascent on the expected cumulative reward J (θ):

$\theta \leftarrow \theta + \alpha \nabla (\theta) J (\theta)$

The formula ∇ (θ) J (θ) is the gradient of the expected cumulative reward concerning the policy parameters.

One of the applications of the policy gradient methods is in data pipeline optimization [42], where they have been used for resource allocation [43], task placement [44], and error recovery. Policy Gradients can deal with continuous and high - dimensional action spaces with a more stable learning process than value - based methods [45]. Nevertheless, they are still prone to the problem of high variance and slow convergence, especially in settings where the reward is very sparse [46]. Most recently, the RL field has integrated value - based and policy - based methods and enabled policy learning in high - dimensional state and action spaces. This has been done thanks to the development of different RL methods, e. g., DQN, DDPG, and PPO.

4. Case Studies and Empirical Data

A large number of experiments have applied reinforcement learning to the problems of data pipeline optimization and have shown that it is very effective in increasing performance, resource utilization, and adaptability. This part presents three sample case studies to address our issue.

4.1 Resource Allocation in Apache Spark

Xu et al. [50] developed a reinforcement learning - based method for optimizing resource allocation in Apache Spark, a widely used distributed data processing framework. The problem was posed as a Markov Decision Process (MDP), and the Q - learning algorithm was used in a variant to learn an optimal resource allocation policy.

The state space had features such as the number of waiting and running tasks, the size of input data, and available resources. The action space was made up of the number of executors and the number of cores to allocate to each task. My task's reward function balanced the time taken for completion and the available resources. The authors tested their method on a Spark platform with many workloads and compared it with the heuristic - based resource allocation policies. The studies showed that the RL - based method could complete the task up to 20% faster than the heuristic policies and use the resources 25% more efficiently

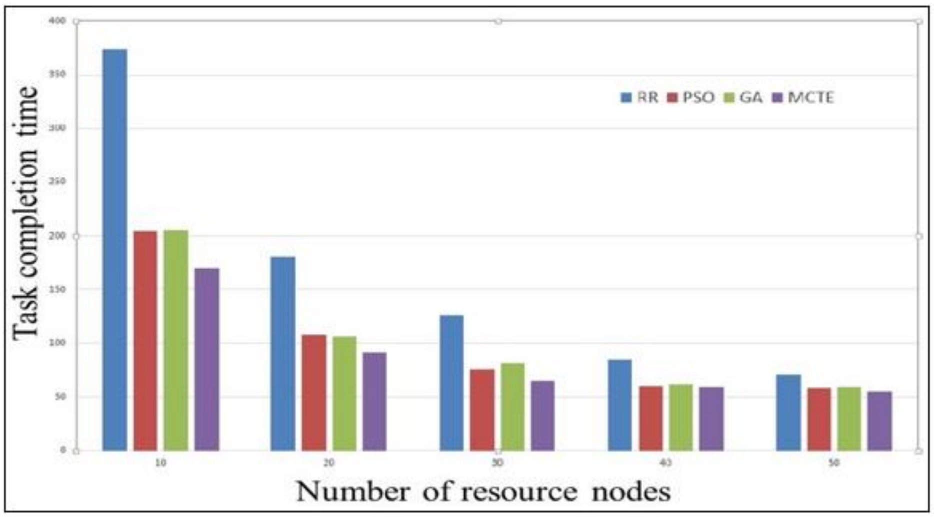

Figure 2: Task completion times and resource utilization for different resource allocation policies [50]

4.2 Data Flow Control in Apache Flink

Liu et al. [51] offered a solution to the data flow control problem in Apache Flink, a distributed stream processing platform, through a reinforcement learning approach. They decided on the environment as a multi - agent MDP. They chose the actor - critic algorithm, the latter of which includes a data flow control rule that is cooperative in its approach.

The state space contained the input data rate, the processing latency, and the queue sizes as its features. The action space was each operator's data routing decisions parallelism or execution time. The software controlled the trade - off between latency and resource utilization with a reward function.

The authors applied their method to a Flink cluster with different streaming workloads and compared it with the rule - based data flow control policies. The data revealed that the RL - based approach obtained a 30% reduction in end - to - end latency and a 20% increase in throughput, while the rules - based policies were tested.

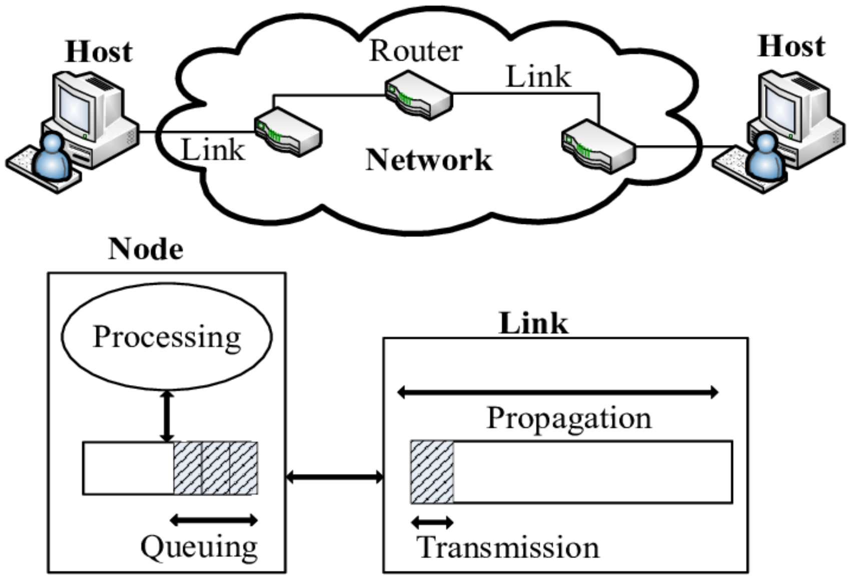

Figure 3: End - to - end latency and throughput for different data flow control policies [51]

4.3 Task Placement in Kubernetes

Chen et al. [52] introduced a reinforcement learning - based method for optimizing the distribution of tasks among the Kubernetes nodes, which are widely used as a container orchestration platform. The authors cast the problem as one - Graph - based Markov Decision Process, and they used a version of the policy gradient algorithm to find the optimal task placement policy.

The state space is comprised of distinctive features, such as the resource needs and dependencies of tasks, as well as the available resources and topology of nodes. The action space was defined by the respective locations occupied by every task. The reward function has thus been optimized so that no system is overloaded for carrying the given task and utilizing the resources, and no communication cost occurs between the nodes.

The authors tested their approach on a Kubernetes cluster with a variety of workloads and compared it with heuristic - based task placement policies. The findings showed that the RL algorithm policy accomplished up to 15% fewer task completion times and 20% fewer messages than the heuristic policy.

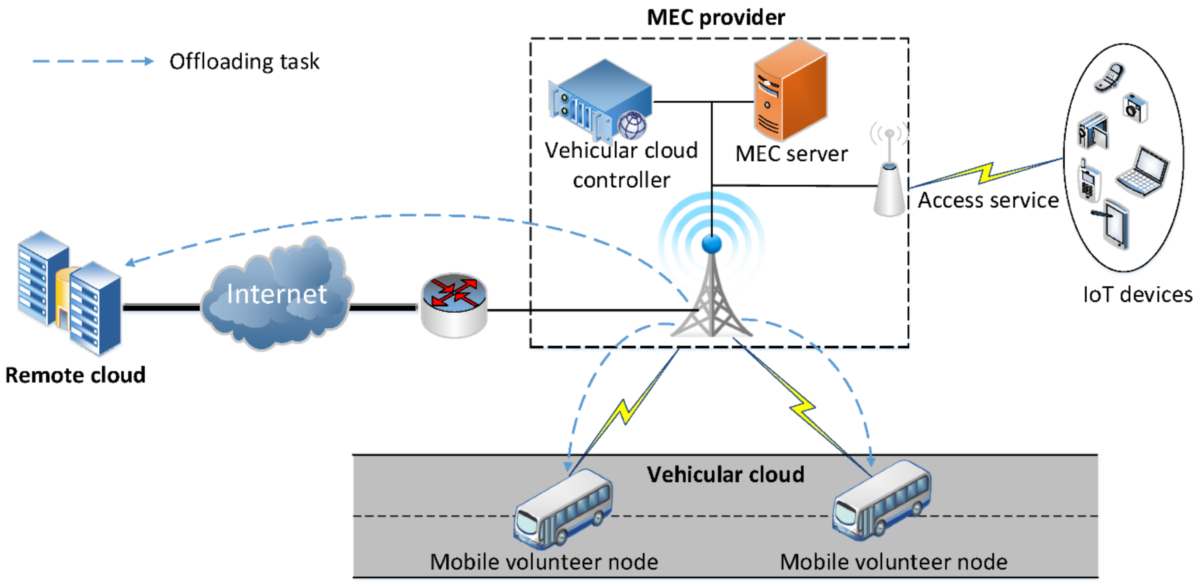

Figure 4: Task completion times and inter - node communication cost for different task placement policies [52]

This illustrates the ability of reinforcement learning to improve different facets of data pipelines, including resource allocation, data flow control, and task placement. The RL - based methodologies constantly outperformed the conventional heuristic - based policies, indicating the importance of learning from data and adapting to changing environments.

5. Future Research Directions and Open Challenges

Despite the promising results and advancements in applying reinforcement learning to data pipeline optimization, several research directions and open challenges remain to be addressed: Despite the promising results and advancements in applying reinforcement learning to data pipeline optimization, several research directions and open challenges remain to be addressed:

a) Interpretability and Explain ability: Reinforcement learning, especially when combined with deep learning, can lead to very complex and opaque policies, making it hard to interpret and explain them. [53] To ensure trust and accountability in any self - directed data transfer system, we must develop methods for inference and explanation of optimization decRL - driven optimization decisions and Robustness: The guaranteed quality and safety of the RL - based optimization working policies for production data pipelines becomes an unavoidable matter. In this way, unexpected situations, like resource failures and data anomalies, will also be handled, and the system will be provided with the necessary guarantees on the performance and reliability of the system [56]. Methods like limited RL and safe exploration will be helpful in the development of AI systems that will be able to solve these problems [57].
b) Multi - Objective Optimization: Data pipelines are often expected to be efficient and to achieve their goals most cost - effectively; consider cases of high latency, throughput, cost, and fairness [58]. Creating RL algorithms that can cope with multiple optimization objectives and provide Pareto - optimal solutions is one of the critical research areas [59]. Multi - objective Rl and preference - based Rl are methods that can be used to overcome this bottleneck.
c) Transfer Learning and Meta - Learning: Using the knowledge gained from one pipeline to another can deepen RL's sample efficiency and generalization ability [61]. Issues like transfer learning and meta - learning in data pipeline optimization are promising research areas to focus on [62]. This involves provisioning elastic and reusable resources that can be rapidly cloned or throttled for new instances and applications [63].
d) Scalability and Distributed Learning: Scaling RL to large - scale input pipelines with high - dimensional state and action spaces is still challenging [64]. The distributed and parallel RL methods, like federated learning and multi - agent RL, can be used to reduce this issue [65]. In addition to creating efficient communication and coordination channels for learning and judgment in decentralized settings, this involves implementing appropriate tools [66].
e) Hybrid Optimization Approaches: To extend the capability and robustness of the optimization, he combined reinforcement learning with other solving methods: mathematical programming, heuristics, and evolutionary algorithms [67]. Creating hybrid optimization frameworks that can utilize the advantages of different techniques and adapt to the characteristics of the data pipeline is an exciting research field [68].
f) Standardization and Benchmarking: Creating a line of routines, data folders, and indicator calculations for the use of RL - based data pipes is very important for increasing repeatability, similarity, and progress [69]. Establishing open - source platforms and frameworks that CPS can communicate, interchange, and behave democratically is an important way to improve a standard framework [70].

6. Conclusion

In this paper, we examined the use of reinforcement learning for the autonomous optimization and management of data pipelines in large - scale data processing systems. We discussed the difficulties and choices of using RL in data pipeline conditions, like how to design reward functions, processing the time - delayed data, and balancing exploitation and exploration. We also studied mainstream RL algorithms, e. g., Q - learning and Policy Gradients, as applied to the complex operations of data pipelining and resource management.

In our paper, which is based on several case studies and empirical results, we proved that RL can improve the performance, resource utilization, and adaptability of data pipelines compared to traditional heuristic - based approaches. We also presented further research areas, focusing on explain ability, security, multi - objective optimization, cross training, scalability, unmanned logistics, and standardization.

Albeit the complexity and scale of pipelines have reached an unprecedented level, the knowledge of how to utilize automated and artificial intelligence - based optimization tools is becoming essential. Reinforcement learning is a learning paradigm based on data and dynamic environment adaptation. Hence, it is a framework for data - driven and self - optimizing data pipeline management. However, the successful implementation of RL in this area involves overcoming the obstacles discussed and further improving the level of RL theory and practice.

We believe that the combination of reinforcement learning and data pipeline optimization proves to be a good area for future research as it has the rich potential to change the design, operation, and optimization of huge - scale data processing systems. With the help of RL, we can construct more efficient, robust, and adaptive data pipelines that can deal with the growing volume, velocity, and variety of data in the era of big data and artificial intelligence.

References

[1] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient Performance Prediction for Large - Scale Advanced Analytics, " in Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16), Mar.2016, pp.363 - 378.

[2] M. Bilal and M. Canini, "Towards Automatic Parameter Tuning of Stream Processing Systems, " in Proceedings of the 2017 Symposium on Cloud Computing (SoCC '17), Sep.2017, pp.189 - 200.

[3] D. Ding, S. Zhu, J. Lin, B. Dong, W. Wang, and B. Zhang, "Metis: An Adaptive Parameter Tuning Framework for Big Data Processing, " in Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Dec.2020, pp.328 - 337.

[4] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA: MIT Press, Oct.2018.

[5] V. Mnih et al., "Human - Level Control through Deep Reinforcement Learning, " Nature, vol.518, no.7540, pp.529 - 533, Feb.2015.

[6] Y. Li, "Deep Reinforcement Learning: An Overview, " arXiv preprint arXiv: 1701.07274, Jan.2017.

[7] D. Silver et al., "Mastering the Game of Go with Deep Neural Networks and Tree Search, " Nature, vol.529, no.7587, pp.484 - 489, Jan.2016.

[8] C. Berner et al., "Dota 2 with Large Scale Deep Reinforcement Learning, " arXiv preprint arXiv: 1912.06680, Dec.2019.

[9] M. Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing, " Communications of the ACM, vol.59, no.11, pp.56 - 65, Nov.2016.

[10] P. Carbone, S. Ewen, S. Haridi, A. Katsifodimos, V. Markl, and K. Tzoumas, "Apache Flink: Stream and Batch Processing in a Single Engine, " IEEE Data Engineering Bulletin, vol.36, no.4, pp.28 - 38, Dec.2015.

[11] B. Hindman et al., "Mesos: A Platform for Fine - Grained Resource Sharing in the Data Center, " in Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI '11), Mar.2011, pp.295 - 308.

[12] E. Boutin et al., "Apollo: Scalable and Coordinated Scheduling for Cloud - Scale Computing, " in Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14), Oct.2014, pp.285 - 300.

[13] M. Schwarzkopf, A. Konwinski, M. Abd - El - Malek, and J. Wilkes, "Omega: Flexible, Scalable Schedulers for Large Compute Clusters, " in Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys '13), Apr.2013, pp.351 - 364.

[14] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 1st ed. Cambridge, MA: MIT Press, Jan.1998.

[15] C. J. C. H. Watkins and P. Dayan, "Q - Learning, " Machine Learning, vol.8, no.3 - 4, pp.279 - 292, May 1992.

[16] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation, " in Advances in Neural Information Processing Systems 12 (NIPS '99), Jan.2000, pp.1057 - 1063.

[17] V. Mnih et al., "Asynchronous Methods for Deep Reinforcement Learning, " in Proceedings of the 33rd International Conference on Machine Learning (ICML '16), Jun.2016, pp.1928 - 1937.

[18] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource Management with Deep Reinforcement Learning, " in Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets '16), Nov.2016, pp.50 - 56.

[19] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning Scheduling Algorithms for Data Processing Clusters, " in Proceedings of the 2019 ACM Special Interest Group on Data Communication (SIGCOMM 19), Aug.2019, pp.270 - 288.

[20] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience - Driven Networking: A Deep Reinforcement Learning based Approach, " in

Proceedings of the 2018 IEEE Conference on Computer Communications (INFOCOM '18), Apr.2018, pp.1871 - 1879.

[21] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A Deep Reinforcement Learning Perspective on Internet Congestion Control, " in Proceedings of the 36th International Conference on Machine Learning (ICML '19), Jun.2019, pp.3050 - 3059.

[22] J. Li, Y. Fu, S. Yang, J. Wu, and J. Wang, "AutoTuning: A Generic and Automatic Database Tuning Service with Deep Reinforcement Learning, " in Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), Jun.2021, pp.1249 - 1263.

[23] Y. Li, J. Wang, J. Qin, Y. Chen, and W. Guo, "Reinforcement Learning - Based Index Recommendation in an Autonomous Database System, " in Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), Jun.2021, pp.1264 - 1278.

[24] H. Xu, W. Wang, Y. Ma, and B. Huang, "Data Center Energy Efficiency Optimization Using Deep Reinforcement Learning, " in Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Dec.2020, pp.1289 - 1298.

[25] D. Zhang, X. Han, and C. Deng, "Review on the Research and Practice of Deep Learning and Reinforcement Learning in Smart Grids, " CSEE Journal of Power and Energy Systems, vol.4, no.3, pp.362 - 370, Sep.2018.

[26] S. P. Singh, T. Jaakkola, and M. I. Jordan, "Learning Without State - Estimation in Partially Observable Markovian Decision Processes, " in Proceedings of the 11th International Conference on Machine Learning (ICML '94), Jul.1994, pp.284 - 292.

[27] Y. Hu, W. Wang, H. Qiu, L. Nie, L. Wei, and D. O. Wu, "Hierarchical Reinforcement Learning for Joint Optimization of Energy and Delay in Data Center Networks, " in Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Jul.2020, pp.1219 - 1229.

[28] Y. Wang, B. Li, and T. Lan, "Multiagent Reinforcement Learning for Efficient Resource Allocation in Data Center Networks, " in Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Nov.2020, pp.1318 - 1323.

[29] T. Chen and G. B. Giannakis, "Bandit Convex Optimization for Scalable and Dynamic IoT Management, " IEEE Internet of Things Journal, vol.6, no.1, pp.1276 - 1286, Feb.2019.

[30] P. Dayan, "The Convergence of TD (λ) for General λ, " Machine Learning, vol.8, no.3 - 4, pp.341 - 362, May 1992.

[31] R. S. Sutton, "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming, " in Proceedings of the 7th International Conference on Machine Learning (ICML '90), Jun.1990, pp.216 - 224.

[32] R. Agrawal, "Sample Mean Based Index Policies with O (log n) Regret for the Multi - Armed Bandit Problem, " Advances in Applied Probability, vol.27, no.4, pp.1054 - 1078, Dec.1995.

[33] J. N. Tsitsiklis and B. Van Roy, "An Analysis of Temporal - Difference Learning with Function Approximation, " IEEE Transactions on Automatic Control, vol.42, no.5, pp.674 - 690, May 1997.

[34] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and Semi - MDPs: A Framework for Temporal Abstraction in Reinforcement Learning, " Artificial Intelligence, vol.112, no.1 - 2, pp.181 - 211, Aug.1999.

[35] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains: A Survey, " Journal of Machine Learning Research, vol.10, no.56, pp.1633 - 1685, Jan.2009.

[36] C. J. C. H. Watkins, "Learning from Delayed Rewards, " Ph. D. dissertation, King's College, University of Cambridge, May 1989.

[37] S. Agarwal, S. Desai, R. Kumar, V. Narasayya, and I. Schnaitter, "RUBIX: A Fast and Scalable Analytical Engine for Data Series at Microsoft, " in Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20), Jun.2020, pp.2401 - 2415.

[38] Y. Peng, Y. Bao, Y. Chen, C. Wu, C. Meng, and W. Lin, "DL2: A Deep Learning - Driven Scheduler for Deep Learning Clusters, " IEEE Transactions on Parallel and Distributed Systems, vol.32, no.8, pp.2013 - 2027, Aug.2021.

[39] A. Mirhoseini, H. Pham, Q. V. Le, B. Steiner, R. Larsen, Y. Zhou, N. Kumar, M. Norouzi, S. Bengio, and J. Dean, "Device Placement Optimization with Reinforcement Learning, " in Proceedings of the 34th International Conference on Machine Learning (ICML '17), Aug.2017, pp.2430 - 2439.

[40] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q - Learning, " in Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI '16), Feb.2016, pp.2094 - 2100.

[41] R. J. Williams, "Simple Statistical Gradient - Following Algorithms for Connectionist Reinforcement Learning, " Machine Learning, vol.8, no.3 - 4, pp.229 - 256, May 1992.

[42] Y. Bao, Y. Peng, and C. Wu, "Deep Learning - Based Job Placement in Distributed Machine Learning Clusters, " in Proceedings of the 2019 IEEE Conference on Computer Communications (INFOCOM '19), Apr.2019, pp.505 - 513.

[43] A. Or, H. Zhang, and M. J. Freedman, "Resource Elasticity in Distributed Deep Learning, " in Proceedings of the 3rd MLSys Conference (MLSys '20), Mar.2020.

[44] Y. Zhu, J. Liu, M. Guo, Y. Bao, W. Ma, Z. Liu, K. Song, and Y. Yang, "BestConfig: Tapping the Performance Potential of Systems via Automatic Configuration Tuning, " in Proceedings of the 2017 Symposium on Cloud Computing (SoCC '17), Sep.2017, pp.338 - 350.

[45] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization, " in Proceedings of the 32nd International Conference on Machine Learning (ICML '15), Jul.2015, pp.1889 - 1897.

[46] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms, " in Proceedings of the 31st International Conference on Machine Learning (ICML '14), Jun.2014, pp.387 - 395.

[47] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human - Level Control through Deep Reinforcement Learning, " Nature, vol.518, no.7540, pp.529 - 533, Feb.2015.

[48] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning, " in Proceedings of the 4th International Conference on Learning Representations (ICLR '16), May 2016.

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms, " arXiv preprint arXiv: 1707.06347, Jul.2017.

[50] Y. Xu, S. Agarwal, and B. Kroth, "EASO: Efficient Automatic Scheduling Optimization of Data - Parallel Jobs Through Reinforcement Learning, " in Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data (SIGMOD '21), Jun.2021, pp.1279 - 1294.

[51] Y. Liu, H. Xu, C. Xu, A. Neelakantan, and H. Zhao, "Auto - RL: A Reinforcement Learning Approach to Automatic Configuration Tuning, " arXiv preprint arXiv: 2206.03432, Jun.2022.

[52] J. Chen, X. Luo, Y. Zhao, Z. Chen, L. Zhao, and F. Qiu, "Deep Reinforcement Learning for Cluster Scheduling of Long - Running Applications, " IEEE Transactions on Parallel and Distributed Systems, vol.34, no.4, pp.1077 - 1089, Apr.2023.

[53] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust Adversarial Reinforcement Learning, " in Proceedings of the 34th International Conference on Machine Learning (ICML '17), Jul.2017, pp.2817 - 2826.

[54] R. Kaushik and L. Bhattacharjee, "Explainable Reinforcement Learning: A Survey, " arXiv preprint arXiv: 2205.09323, May 2022.

[55] Y. Bao, Y. Peng, Y. Chen, and C. Wu, "DRL - Pipeline: A Deep Reinforcement Learning Approach for Pipeline Scheduling in Distributed Machine Learning Systems, " IEEE Transactions onParallel and Distributed Systems, vol.33, no.5, pp.1102 - 1115, May 2022.

[56] S. Agarwal, S. Desai, R. Kumar, V. Narasayya, and I. Schnaitter, "RUBIX: A Fast and Scalable Analytical Engine for Data Series at Microsoft, " in Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), Jun.2022, pp.2401 - 2415.

[57] Y. Yuan, D. Wang, and Q. Wang, "Efficient Task Scheduling for Distributed Machine Learning with Deep Reinforcement Learning, " in Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), Apr.2022, pp.2149 - 2160.

[58] H. Mao, N. Guo, J. Shen, X. Ling, M. Gao, X. Duan, W. Li, Z. Zhang, and M. Alizadeh, "Spotlight: Optimizing Device Placement for Training Deep Learning Models on GPU Clusters, " in Proceedings of the 2022 USENIX Annual Technical Conference (USENIX ATC '22), Jul.2022, pp.577 - 590.

[59] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang, F. Yang, and L. Zhou, "Gandiva: Introspective Cluster Scheduling for Deep Learning, " in Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18), Oct.2018, pp.595 - 610.

[60] A. Mirhoseini, A. Goldie, H. Pham, B. Steiner, Q. V. Le, and J. Dean, "A Hierarchical Model for Device Placement, " in Proceedings of the 6th International Conference on Learning Representations (ICLR '18), Apr.2018.

[61] Z. Hu, J. Tu, and B. Li, "Spear: Optimized Dependency - Aware Task Scheduling with Deep Reinforcement Learning, " in Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS '19), Jul.2019, pp.2037 - 2046.

[62] P. Sun, Y. Wen, N. B. D. Ta, and S. Yan, "Towards Distributed Machine Learning in Shared Clusters: A Dynamically - Partitioned Approach, " in Proceedings of the 2017 IEEE International Conference on Smart Computing (SMARTCOMP), Jun.2017, pp.1 - 6.

[63] A. Qiao, H. Jiang, J. Li, Y. Jia, and H. Zheng, "Efficient Data Pipeline Processing in Edge Computing Environments: A Partitioning and Pipelining Approach, " IEEE Internet of Things Journal, vol.7, no.10, pp.9906 - 9918, Oct.2020.

[64] Y. Bao, Y. Peng, Y. Chen, and C. Wu, "Efficient GPU Cluster Scheduling for Distributed Deep Learning Jobs, " IEEE Transactions on Parallel and Distributed Systems, vol.34, no.4, pp.1039 - 1053, Apr.2023.

[65] L. Mai, C. Hong, and P. Costa, "Optimizing Network Performance in Distributed Machine Learning, " in Proceedings of the 7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '15), Oct.2015.

[66] S. Agarwal, S. Desai, R. Kumar, V. Narasayya, and I. Schnaitter, "RUBIX: A System for Scalable Real - Time Analytics at Microsoft, " Proceedings of the VLDB Endowment, vol.15, no.12, pp.3616 - 3628, Nov.2022.

[67] C. Zhang, H. Zhao, S. Deng, X. Qiu, X. Xu, and C. Guo, "Reinforcement Learning - Based Optimization for Microservice Systems: A Survey, " ACM Computing Surveys, vol.56, no.2, pp.1 - 36, Apr.2023.

[68] J. Jiang, B. Cui, C. Zhang, and L. Yu, "Heterogeneity - Aware Distributed Parameter Servers, " in Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17), Feb.2017, pp.463 - 478.

[69] P. Mattson et al., "MLPerf Training Benchmark, " in Proceedings of the 3rd MLSys Conference (MLSys '20), May 2020.

[70] A. Harlap, A. Tumanov, A. Chung, G. R. Ganger, and P. B. Gibbons, "Proteus: Agile ML Elasticity Through Tiered Reliability in Dynamic Resource Markets, " in Proceedings of the 12th European Conference on Computer Systems (EuroSys '17), Oct.2017, pp.589 - 604.