Poster Abstract: Configuration Tuning for Distributed IoT Message Systems Using Deep Reinforcement Learning

Zhuangwei Kang*
Vanderbilt University
Nashville, Tennessee
zhuangwei.kang@vanderbilt.edu

Yogesh D. Barve
Vanderbilt University
Nashville, Tennessee
yogesh.d.barve@vanderbilt.edu

Shunxing Bao
Vanderbilt University
Nashville, Tennessee
shunxing.bao@vanderbilt.edu

Abhishek Dubey
Vanderbilt University
Nashville, Tennessee
abhishek.dubey@vanderbilt.edu

Aniruddha Gokhale
Vanderbilt University
Nashville, Tennessee
a.gokhale@vanderbilt.edu

ABSTRACT

Distributed messaging systems (DMSs) are often equipped with a large number of configurable parameters that enable users to define application run-time behaviors and information dissemination rules. However, the resulting high-dimensional configuration space makes it difficult for users to determine the best configuration that can maximize application QoS under a variety of operational conditions. This poster introduces a novel, automatic knob tuning framework called DMSConfig. DMSConfig explores the configuration space by interacting with a data-driven environment prediction model(a DMS simulator), which eliminates the prohibitive cost of conducting online interactions with the production environment. DMSConfig employs the deep deterministic policy gradient (DDPG) method and a custom reward mechanism to learn and make configuration decisions based on predicted DMS states and performance. Our initial experimental results, conducted on a single-broker Kafka cluster, show that DMSConfig significantly outperforms the default configuration and has better adaptability to CPU and bandwidth-limited environments. We also confirm that DMSConfig produces fewer violations of latency constraints than three prevalent parameter tuning tools.

CCS CONCEPTS

• **Software and its engineering → Software configuration management and version control systems**; • **Computing methodologies → Policy iteration**.

KEYWORDS

Publish/Subscribe Middleware, System Configuration, Policy-based RL Algorithm

ACM Reference Format:
Zhuangwei Kang, Yogesh D. Barve, Shunxing Bao, Abhishek Dubey, and Aniruddha Gokhale. 2021. Poster Abstract: Configuration Tuning for Distributed IoT Message Systems Using Deep Reinforcement Learning. In *International Conference on Internet-of-Things Design and Implementation (IoTDI '21), May 18–21, 2021, Charlottesvle, VA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3450268.3453517

IoTDI '21, May 18–21, 2021, Charlottesvle, VA, USA

ACM ISBN 978-1-4503-8354-7/21/05.
https://doi.org/10.1145/3450268.3453517

1 INTRODUCTION

A variety of IoT application domains, such as smart cities and smart grids, employ DMS as the middleware for data transmission through which messages can be produced, disseminated and consumed asynchronously. To ensure flexibility in a wide range of deployment scenarios, system topologies and runtime specifications, industrial-strength DMSs provide users with a set of continuous and discrete configurable parameters that have different data types (e.g., numeric, boolean, categorical) and value ranges, which together result a hybrid, multidimensional configuration space. These parameters control application runtime behaviors and resource allocation strategies, resulting in different variations of application performance measured across different metrics, such as throughput, latency, CPU utilization, etc.

Making prudent configuration decisions is challenging because the scale of the searching space boosts exponentially as the quantity of tunable parameters increases. It also requires significant domain knowledge and in-depth understanding of the impact of each parameter on application performance as well as their unseen interactions, which is difficult even for experts, not to mention common users. The default configurations provided by the software vendors are usually suboptimal, and utilizing naïve exhaustive search methods to find appropriate configurations are laborious, time-consuming, non-scalable, and likely be suboptimal due to the continuous nature of the configuration space (most parameters are numeric). Hence, this project proposes a Deep Reinforcement Learning (DRL)-based configuration recommendation system, called DMSConfig. We aim to optimize the publisher-side throughput of DMS applications while meeting latency constraints, which satisfies the demands of practical IoT streaming applications, such as online smart grid analytics, that usually have stringent requirements on both throughput and response time. This project uses Kafka, a popular event-processing framework used at the data analytics layer of IoT systems, as an example to validate our approach, but DMSConfig can be adapted to other DMSs since its system components are fully decoupled.

2 METHODOLOGY

DMSConfig is built using container-based emulation techniques, conventional machine learning, and the DDPG [5] DRL-based algorithm, which is utilized in three stages (Figure 1): data collection, DMS simulator training, and configuration tuning. The data collection phase includes two steps: sampling configurations and executing them on a testbed. To expose the relationship between parameter combinations and performance as much as possible within a limited operational budget, we employ the Latin Hypercube Sample (LHS) [4] method to guarantee that the training set covers the configuration space uniformly. Next, we leverage container and traffic control techniques to emulate practical DMS production environments in terms of CPU, memory, and bandwidth, which enhance the system adaptivity. The resource isolation capabilities of container virtualization aid in parallelizing the configuration evaluation process thereby reducing the cost of data collection.

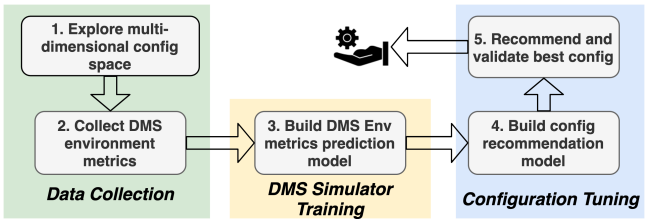


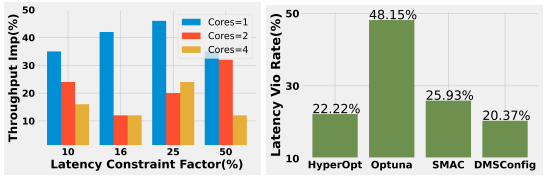
Figure 1: DMSConfig Workflow and Architecture

In the second stage, DMSConfig adopts the random forest (RF) algorithm to train a DMS simulator that takes a number of performance-relevant parameters as input and forecasts several software internal state metrics, publisher-side throughput, and latency. The use of the DMS simulator enables the parameter tuner in the next step to obtain a predicted environmental feedback immediately after making a configuration decision, so that the tuner can locate the high-quality configurations within an acceptable time budget.

In the third phase, we convert the latency-constrained configuration tuning problem to a Markov Decision Process and solve it using the DDPG algorithm. Briefly, the auto-tuner (RL Agent) gradually enhances the likelihood of selecting high-quality configurations (RL Action) through trial and error. The derived optimal searching strategy (RL Policy) can navigate the auto-tuner to obtain the maximum cumulative return, and the action taken to reach the terminal state is the best configuration. The benefit of DDPG is: (1) the sequential decision-making process in RL is akin to iterative parameter adjustment; (2) DDPG has been proven to be a robust approach for settling continuous control problems (continuous configuration in our context); (3) the reward function in RL guides the tuning process by applying revenue or penalty to the agent, which satisfies our demand for throughput and latency simultaneously; (4) driven by the model-based DMS simulator and RL reward mechanism, DMSConfig can rapidly adapt tuning requests that have different latency constraints.

3 RESULTS AND CONCLUSIONS

Our ongoing work evaluates DMSConfig under 9 Kafka use cases (i.e., different number of publishers, CPU cores, bandwidth, and message size) and comparing its performance with three mature



(a) T_{imp} **vs.** lcf **(b)** lvr **vs.** $optimizer$

Figure 2: (a) Percentage of throughput improvement offered by DMSConfig over the default configuration under different number of publisher CPU cores and lcf; (b) Overall latency violation rate under 7 levels of lcf over 9 use cases.

hyper-parameter recommendation tools (HyperOpt[2], Optuna[1], and SMAC[3]) and the Kafka's default configuration. Besides, we exam 7 different levels (10, 12, 16, 25, 50, 100, unlimited) of latency constraint factors (lcf), where lcf is the percentage of the default configuration latency. Each lcf) is used to validate whether our proposed DMSConfig can successfully balance latency and throughput and obtain maximum profit. Our preliminary results reveal that the configurations identified by DMSConfig achieve overall 12%-538% throughput improvement under 7 different levels of lcf. DMSConfig is also able to guarantee application performance under resource-constrained environments by making effective configuration recommendations. In a 100Mbps bandwidth environment, for instance, it promotes throughput by 463%-538% under the premise of satisfying 7 levels of lcf. Likewise, vertical comparison results shown in figure 2a prove DMSConfig adapts to diverse publisher CPU core settings(show 4 lcf due to page limit constraint). Preliminary results also confirm that DMSConfig earns analogous throughput performance compared with the three baselines but delivers the most reliable latency guarantees (see Figure 2b). In conclusion, DMSConfig illustrates a promising approach for configuration tuning problems. To further improve the DMSConfig performance, in future we plan to (1) optimize the DDPG reward function and neural network design to enhance throughput and reduce latency violation occurrence rate; (2) extend the single-broker DMS configuration problem to multi-broker scenarios.

ACKNOWLEDGMENTS

This work is supported by a funding from Cisco. Any opinions, findings, and conclusions or recommendations expressed in this material are of the author(s) and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2623–2631.

[2] James Bergstra, Dan Yamins, and David D Cox. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference*, Vol. 13. Citeseer, 20.

[3] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*. Springer, 507–523.

[4] Ronald L Iman. 2014. L atin hypercube sampling. *Wiley StatsRef: Statistics Reference Online* (2014).

[5] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).