Poster: Configuration Management for Internet Services at the Edge: A Data-Driven Approach

Yue Zhang
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH, USA
zhang.8016@osu.edu

Christopher Stewart
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH, USA
cstewart@cse.ohio-state.edu

***Abstract*—Internet services are increasingly pushed from the remote cloud to the edge sites close to data sources to offer fast response time and low energy footprint. However, software deployed at edge sites must be updated frequently. Performing updates as soon as they are available consumes a large amount of energy. Configuration management tools that install software updates and manage allowed staleness can inflate energy demands, especially when updates interrupt idle periods at the edge site and block processors from entering power-saving modes. Our research studies configuration management policies, their effect on energy footprint and strategies to optimize them. We have observed that policies yielding low energy footprint differ from site to site and over time. We propose a data-driven approach that uses data collected at each edge site to predict an energy-efficient policy and also guards against worst-case performance if data-driven predictions error occurs. We use a novel random-walk approach to manage data-driven policies that yield a low footprint for a representative trace of updates observed at an edge site. We are setting up 4 edge service benchmarks powered by AI inference to create realistic software update traces.**

***Index Terms*—data-driven, Internet service at edge, AI on IoT**

I. Introduction

Internet services are increasingly deployed at edge sites to provide low-latency responses thereby reducing energy footprint [13]. An edge site is a low-power device, with processing capability that allows multiple Internet services deployed at the proximity of the edge site. The low energy footprint of edge sites makes them ideal solutions to deploy different workloads that use machine learning (ML) [16]. The applications have ML models pack-aged as software and deployed on the IoT device connected to an edge device. Examples include smart cameras, smartwatches, smart traffic controls, drones, and so on [3]–[5], [7], [9], [11], [15], [20], [21]. These devices sense surroundings and provide local inference or relay requests to edge sites running internet service.

The software deployed for such internet services must be updated regularly in order to avoid data drift [14] and to improve the performance of internet service. Failure to update the software would result in poor internet service which provides inaccurate and inefficient inferences. When software deployed are allowed to stale, it becomes a challenging problem to perform software updates by consuming minimal energy. Edge sites can enter power saving mode if they decide to skip updating.

Aggressive updating policies that perform the updates as soon as they become available to avoid staleness violation consume enormous energy. The most conservative approach of delaying updates could trap the device from entering deep power-saving mode when many updates arrive that lead to continuous staleness violation. With multiple software updates, deciding whether to skip updates or how many updates to perform and in what order to perform updates would be a challenging problem to tackle. Given the trace in advance, the complexity of an offline solution to such a problem is NP-Hard [2].

Given a trace of updates such that factors do not change, there exist techniques such as random walks [2] or reinforcement learning [1] in order to find the best scheduling policy to perform software updates in an energy-efficient manner. Continuing with fixed factors for a period of time, such a scheduling policy performs updates consuming minimal energy.

In reality, there are multiple factors that affect a policy performing software updates such as update rate, allowed staleness, number of applications deployed, idle time length, bandwidth, and size of the update [6]. Given a historic trace of updates with fixed factors, one could find an energy-efficient updating policy offline and apply the policy to updates that arrive in the future. Changes to these factors worsen the policy, forcing it to consume excess energy. The policy needs to be updated in order to be energy-efficient. We do not know these factors in advance and subtle changes to these factors affect the currently preferred software scheduling policy. Over time, when such factors change and worsen the energy consumption, there should be changed to the current update policy or fall back to a safe updating policy that can sustain the demand and factors better than the current update policy. Such policies differ from site to site and each edge site requires a configuration management tool to perform energy-efficient software updates.

Our research proposes data-driven configuration management for edge sites. Configuration management, inspired solely by data collected from edge sites, predicts an effective policy to perform software updates. Configuration manage-

ment not only encourages multiple policies that perform energy-efficient updates but guards against the worst-case when such policies worsen due to change in factors and data-driven predictions error [12]. Configuration management periodically collects traces and performs random walk to select energy-efficient software updating policy. We build configuration management and implement a data-driven approach to perform software updates.

We test data-driven configuration management using the simulation of realistic internet services for different workloads and edge devices. The workloads include image detection on Mnist(MNIST), intruder detection(Intrude), traffic sign recognition(TSR), and human activity recognition(HAR). The edge site could be HP-Laptop, raspberry-pi with the provision of adding different accelerators on top of them for faster inference. We compare our data-driven approach to state-of-the-art inspired systems such as GAIA [8], SCEDA [1], which proposed a model synchronization mechanism that ensures staleness by reinforcement learning. Data-Driven configuration management achieves energy footprint much lower than first come first serve (FCFS) scheduling for all workloads, and a little bit higher than the offline optimal.

II. Design

AI-Driven IoT encompasses AI applications that are deployed on edge computing platforms for fast inference and low-latency results. The Applications have ML models packaged as software and deployed on the IoT device connected to an edge device. Multiple such edge devices could serve numerous IoT devices with AI processing capability. Figure 1 depicts the AI-Driven IoT workflow. ML models are trained on cloud servers with sufficient compute capability and pushed to edge devices for updates. Edge devices perform these updates according to the scheduler deployed.

[2] suggests that given the option of performing updates or putting edge device to sleep in an inter-arrival period, the update order impacts the amount of energy incurred for processing updates.

However, given the option to choose between static (context-aware) scheduler and a dynamic scheduler (which can adapt to context changes), we need to strike a balance of picking scheduler over period of time. We do not know when context changes and reverts back, number of updates, inter-arrival rate in advance. This research focuses on implementing an online scheduler which picks correct scheduler in order to minimize energy cost to perform application updates.

In order to solve the problem in hand, we might think of deploying a static context-aware scheduler or dynamic adaptive scheduler for entirety. However, deploying a static context aware scheduler or adaptive scheduler for entirety would be a costly operation. It consumes orders of magnitude of additional energy as context changes happen often in practice. [10] We must devise an online algorithm which has good competitive guarantees and perform well in practice. Here, we are inspired by prior work using online algorithms in systems management [17]–[19].

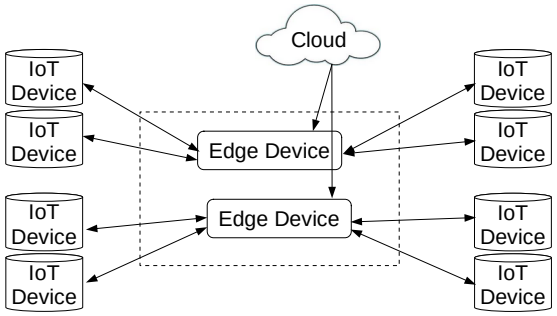

Fig. 1. AI-Driven IoT

To deploy our contribution, we design two schedulers: 1) context-aware scheduler which has learned the context from past traces and 2) context adaptive scheduler which performs updates in a first come first server (FCFS) manner. There should be context configuration scripts which specifies a time-stamped context changes over period of time. The updates are pushed and are held at the edge device in a queue. The edge device processes inference and during idle time it processes the updates or puts the device to deep sleep. The scheduler deployed at the edge is very crucial in order to minimize the energy required to process the updates. The online scheduler switches schedulers for every inter-arrival period by deciding to pick context-aware versus choosing context adaptive scheduler.

With this deployment and running a trace, one could get energy to process the updates using offline optimal, online scheduler and default context-adaptive scheduler.

References

[1] A. Aral, M. Erol-Kantarci, and I. Brandić. Staleness control for edge data analytics. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2):1–24, 2020.

[2] N. T. Babu and C. Stewart. Energy, latency and staleness tradeoffs in ai-driven iot. In *Symposium on Edge Computing*, 2019.

[3] S. Bhattacharya and N. D. Lane. From smart to deep: Robust activity recognition on smartwatches using deep learning. In *Pervasive Computing and Communication Workshops*, 2016.

[4] J. Boubin, N. Babu, C. Stewart, J. Chumley, and S. Zhang. Managing edge resources for fully autonomous aerial systems. In *ACM Symposium on Edge Computing*, 2019.

[5] J. Boubin, J. Chumley, C. Stewart, and S. Khanal. Autonomic computing challenges in fully autonomous precision agriculture. In *IEEE International Conference on Autonomic Computing*, 2019.

[6] J. Chen and X. Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[7] S. Flutura, A. Seiderer, I. Aslan, C. T. Dang, R. Schwarz, D. Schiller, and E. Andre. Drinkwatch: A mobile wellbeing application based on interactive and cooperative machine learning. In *International Conference on Digital Health*, 2018.

[8] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu. Gaia: Geo-distributed machine learning approaching LAN speeds. In *Symposium on Networked Systems Design and Implementation*, 2017.

[9] M. A. Kader, E. Bastug, M. Bennis, E. Zeydan, A. Karatepe, A. S. Er, and M. Debbah. Leveraging big data analytics for cache-enabled wireless networks. In *IEEE Globecom Workshops (GC Wkshps)*, 2015.

[10] M. Kim, J. Lee, Y. Kim, and Y. H. Song. An analysis of energy consumption under various memory mappings for fram-based iot devices. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, 2018.

[11] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar. Squeezing deep learning into mobile and embedded devices. *IEEE Pervasive Computing*, 16(3):82–88, 2017.

[12] R. Lee, M. H. Hajiesmaili, and J. Li. Learning-assisted competitive algorithms for peak-aware energy scheduling. In *arXiv preprint arXiv:1911.07972*, 2019.

[13] P. Liu, D. Willis, and S. Banerjee. Paradrop: Enabling lightweight multitenancy at the network's extreme edge. In *ACM Symposium on Edge Computing*, 2016.

[14] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. Dataset shift in machine learning. MIT Press, 2009.

[15] F. Shahmohammadi, A. Hosseini, C. E. King, and M. Sarrafzadeh. Smartwatch based activity recognition using active learning. In *International Conference on Connected Health: Applications, Systems and Engineering Technologies*, 2017.

[16] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen. Convergence of edge computing and deep learning: A comprehensive survey. In *IEEE Communications Surveys & Tutorials*, 2020.

[17] Z. Xu, N. Deng, C. Stewart, and X. Wang. Cadre: Carbon-aware data replication for geo-diverse services. In *IEEE International Conference on Autonomic Computing*, 2015.

[18] Z. Xu, C. Stewart, N. Deng, and X. Wang. Blending on-demand and spot instances to lower costs for in-memory storage (winner best-in-session presentation). In *IEEE International Conference on Computer Communications*, 2016.

[19] Z. Xu, C. Stewart, and J. Huang. Elastic, geo-distributed raft. In *ACM International Symposium on Quality of Service*, 2019.

[20] S. Yi, Z. Hao, Z. Qin, and Q. Li. Fog computing: Platform and applications. In *Hot Topics in Web Systems and Technologies (HotWeb)*, 2015.

[21] S. Zhang and C. Stewart. Computational thinking curriculum for unmanned aerial systems. In *IEEE National Aerospace and Electronics Conference*, 2019.