Revisiting Online Scheduling for AI-Driven Internet of Things

Naveen T.R. Babu and Christopher Stewart, The Ohio State University

ABSTRACT

AI-driven Internet of Things (IoT) use AI Inference to characterize data processed from various sensors. Together, AI and IoT support smart buildings, cities, cars and drones. However, AI Inference requires updates when executed in new contexts. Frequent updates consume more energy and drain precious IoT batteries. Updates can be batched together to save energy, but it is challenging to batch updates well without knowing when updates will arrive, what their processing needs will be and how long they can be delayed. This work studies update batching and its potential energy savings. We define an update batching policy as a sequence of discrete choices about when to apply concurrent updates. This allows us to use random walks to sample update batching policies. Random walks simulate nearly 1M batching policies and models their energy footprint for an AI-driven IoT comprising 50 AI Inference components. The best policy uses much less energy than 99^{th} and 95^{th} percentiles. First-come-first-serve and Shortest-job-first policies perform like the median sampled batching policy, using 7X more energy.

ACM Reference Format:
Naveen T.R. Babu and Christopher Stewart, The Ohio State University. 2019. Revisiting Online Scheduling for AI-Driven Internet of Things. In *The Fourth ACM/IEEE Symposium on Edge Computing (SEC 2019), November 7–9, 2019, Arlington, VA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3318216.3363326

1 MOTIVATION

Internet of Things (IoT) can sense their surroundings and push data to the cloud for analysis [16]. Increasingly, IoT can execute AI inference using convolutional neural networks [12], deep random forests [17], etc. AI inference characterizes sensed data, reduces bandwidth to the cloud and enables low latency reactions [16].

There are many paradigms for IoT to use AI inference. For example, IoT could use onboard processors to (1) harvest data, (2) train AI models and (3) execute AI inference. In this paper, we study a competing model where IoT download models trained in the cloud. This paradigm fits nicely with power constrained processors on modern IoT. Also, this paradigm allows AI models to train with data from diverse sources, improving accuracy. Early examples of AI-driven IoT include (1) the AI Developer Kit [3] that pairs Qualcomm Neural Acceleration and Microsoft AI for smart cameras and (2) Fully Autonomous Precision Agriculture where smart drones use AI inference to sample large fields for crop scouting [5, 6].

Problem Statement: Deep learning models are produced from training data and model parameters. When either of these inputs change, IoT must update their AI software or risk inaccurate results. For example, in a common practice called progressive sampling, cloud data centers initially push truncated model parameters to IoT and progressively push full parameters over time. This approach trades classification accuracy for latency [13, 16].

If IoT employ few and small AI inference models, updates caused by progressive sampling and model retraining have negligible resource demands. However, as AI-driven IoT proliferate, the aggregate energy footprint of updates can drain IoT batteries and reduce their lifetime. *When updates are available, IoT schedulers have two choices: delay or install now.* It helps to delay updates because, if they are installed in batches, IoT processors enjoy long idle periods, transition to deep sleep modes and save energy. However, delaying updates degrades accuracy. Prior work has shown that staleness corresponds to answer quality [7, 9, 10]. Hard limits on staleness prevent gross degradation on quality. However, even with staleness bounds, deciding when to install updates for energy efficiency is challenging since update arrival times and their processing needs are stochastic.

An IoT scheduler repeatedly chooses to delay or apply available updates. An *update batching policy* is the collection of choices over a trace of update arrivals. Figure 1 uses fully autonomous aerial systems [6], an emerging class of AI-driven IoT, to depict batching policies. With fully autonomous aerial systems, unmanned aircraft are piloted wholly by software. First, AI path planning algorithms specify a flight action. When the action completes, the aircraft takes a picture and uses AI inference to characterize its surroundings and choose the next flight action. While the aircraft flies, processors idle. During this time, AI software can be updated or processors can enter deep sleep mode.

SEC 2019, November 7–9, 2019, Arlington, VA, USA

ACM ISBN 978-1-4503-6733-2/19/11…$15.00
https://doi.org/10.1145/3318216.3363326

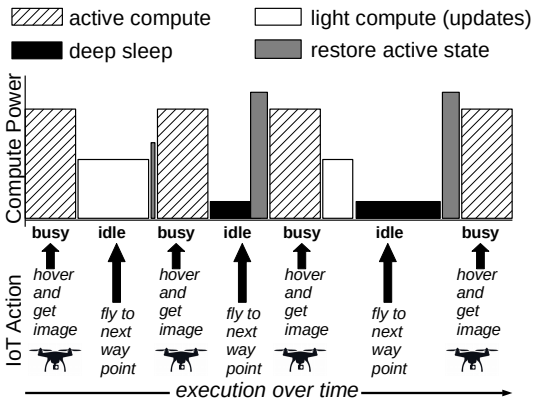

Figure 1: Sample Workload

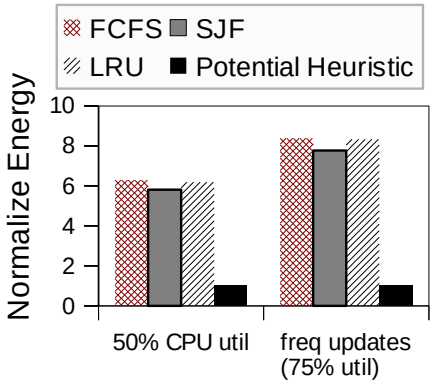

Figure 2: Results

If IoT enter deep sleep, there is a penalty to restore active state. The IoT scheduler seeks to apply AI updates in batches (at light compute power) to free up long idle periods that warrant deep sleep. This problem formulation represents a challenging offline scheduling problem (NP-Hard). In practice, the IoT scheduler must make choices online, making the problem even harder. Below, we outline several important contributions needed for AI-driven IoT.

1. **Online scheduling under high CPU utilization:** As tasks, AI updates are background jobs with limited delay tolerance. The foreground IoT jobs (e.g., image processing and AI inference) must receive priority. As updates become frequent, resource demand and capacity converge. An online scheduling approach that provides competitive guarantees would be a valuable contribution.
2. **Staleness, latency and other workload trade-offs:** Quality, latency and throughput have complex trade-offs [4]. If an IoT scheduler could expose these trade-offs, end users could make personalized choices about device lifetime and performance.
3. **IoT hardware and AI optimization:** We suspect that energy used on AI updates will vary depending on IoT hardware (CPU versus GPU) and AI inference models (CNN versus random forest). For a given workload, standard scheduling policies may perform better under certain hardware and AI combinations. We would like to devise approaches to characterize these interactions.

2 EARLY RESULTS

Our first objective is to characterize the potential impact of scheduling in terms of energy savings. In this early work, we have sampled batching policies by modeling the problem with random walk techniques [14]. Given a trace of update arrival and idle time period, a random walk uses stochastic processes to choose between install now or delay.

Our approach requires a trace of representative updates to AI models. Even though progressive sampling and periodic retraining are common practices, we are unaware of extant traces. We re-purposed energy demands, storage size, transfer time and accuracy data from multiple DNN configurations [13]. We organized DNN models by size and simulated progressive DNN transfer by streaming smaller first and then larger configurations from server to edge. For retraining, we assumed Poisson inter arrivals. This resulting trace produces a sequence of AI model updates with their corresponding (absolute) arrival times and processing requirements. For this paper, we allowed a scheduler to delay updates until a model was 2 versions out of date. In future work, we plan to explore staleness tradeoffs.

We conducted roughly 1M random walks and report results from the policy that provided lowest energy footprint, i.e, *potential heuristic schedule*. We compared that result to traditional online scheduling approaches [15]:

- - First-Come-First-Serve (FCFS),
- - Shortest-Job-First (SJF) and
- - Least-Recently-Updated (LRU)

Figure 2 shows simulation results with IoT composed from 50 AI inference components. Traditional approaches consume 6-8X more energy compared to the best random walk. This suggests there is substantial room for efficient schedulers.

We also computed 99th, 95th and 50th percentiles from our random walk simulations. We found that significant gap exists between the best policy and high ranking policies. For example, we found that 99th percentile can use 1.67X more energy than the best policy. The median policy performed

comparably to traditional scheduling techniques. These results suggest that AI-driven IoT schedulers demand theoretical guarantees that they approximate the best policy. Subtle policy shifts can cost 2X energy usage. In future work, we are further exploring these early findings across hardware and AI profiles. If they stand, AI-driven IoT may usher a new era where scheduling policies are mission critical for the lifetime of IoT and cyber-physical systems.

3 DISCUSSION

Currently, Artificial Intelligence (AI) in edge computing is still in rudimentary stage and there's enormous research going on in this area [1] [11]. We believe as the use of accelerators in edge computing become common, there would be more AI Inference components used in IOT devices and these AI Inference components receive frequent cloud-edge updates over period of time [2]. There is room for implementing an Online scheduling policy for AI-Driven IoT. Traditional Online scheduling techniques like FCFS [15], SJF [15] and LRU [15] aren't energy efficient. From the schedules collected through Random walk, there is significant gap between best and high ranking scheduling policy. Although these are early results, there is scope to perform in-depth Design of experiments [8] to understand the impact of different factors affecting update scheduling and a need to design an efficient Online scheduler for AI-Driven IoT.

Acknowledgments: This work was funded in part by NSF Grants 1749501 and 1350941 with support from NSF CENTRA collaborations (grant 1550126).

REFERENCES

[1] Ai on the edge: Is it ready for prime time? https://www.forbes.com/sites/forbestechcouncil/2019/03/04/ai-on-the-edge-is-it-ready-for-prime-time/, 2019.

[2] How ai accelerators are changing the face of edge computing. https://www.forbes.com/sites/janakirammsv/2019/07/15/how-ai-accelerators-are-changing-the-face-of-edge-computing/, 2019.

[3] Vision ai development kit. https://developer.qualcomm.com/hardware/vision-ai-development-kit, 2019.

[4] Naveen T.R. Babu and Christopher Stewart. Energy, latency and staleness tradeoffs in ai-driven iot. In *ACM Symposium on Edge Computing*, 2019.

[5] Jayson Boubin, Naveen T.R. Babu, John Chumley Christopher Stewart, and Shiqi Zhang. Managing edge resources for fully autonomous aerial systems. In *ACM Symposium on Edge Computing*, 2019.

[6] Jayson Boubin, John Chumley, Christopher Stewart, and Sami Khanal. Autonomic computing challenges in fully autonomous precision agriculture. In *International Conference on Autonomic Computing*, 2019.

[7] Yuxiong He, Sameh Elnikety, and Hongyang Sun. Tians scheduling: Using partial processing in best-effort applications. In *International Conference on Distributed Computing Systems*, 2011.

[8] Raj Jain. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley professional computing. Wiley, 1991.

[9] Jaimie Kelley, Christopher Stewart, Nathaniel Morris, Devesh Tiwari, Yuxiong He, and Sameh Elnikety. Measuring and managing answer quality for online data-intensive services. In *IEEE International Conference on Autonomic Computing*, 2015.

[10] Jaimie Kelley, Christopher Stewart, Nathaniel Morris, Devesh Tiwari, Yuxiong He, and Sameh Elnikety. Obtaining and managing answer quality for online data-intensive services. In *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 2017.

[11] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, and Fahim Kawsar. An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In *Proceedings of the 2015 International Workshop on Internet of Things Towards Applications*, 2015.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553), 2015.

[13] Sicong Liu, Yingyan Lin, Zimu Zhou, Kaiming Nan, Hui Liu, and Junzhao Du. On-demand deep model compression for mobile devices: A usage-driven model selection framework. In *ACM MobiSys*, 2018.

[14] Lovasz and Laszlo. Random walks on graphs: A survey, combinatorics, paul erdos is eighty. *Bolyai Soc. Math. Stud.*, 2:1–46, 01 1993.

[15] Abraham Silberschatz, Greg Gagne, and Peter B Galvin. *Operating system concepts*. Wiley, 2018.

[16] M. Song, K. Zhong, J. Zhang, Y. Hu, D. Liu, W. Zhang, J. Wang, and T. Li. In-situ ai: Towards autonomous and incremental deep learning for iot systems. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018.

[17] Zhi-Hua Zhou and Ji Feng. Deep forest. *arXiv preprint arXiv:1702.08835*, 2017.