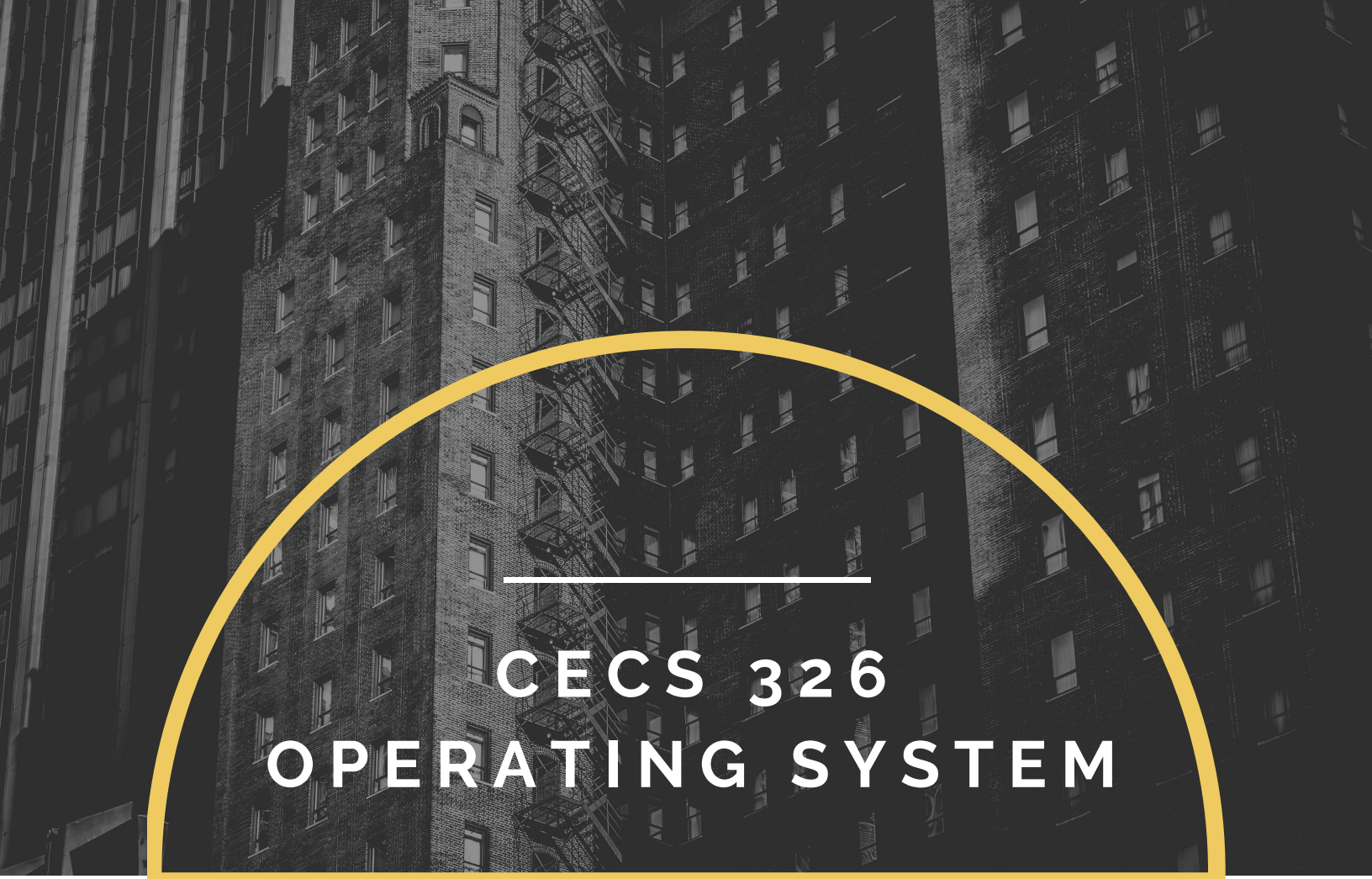

# CECS 326
# OPERATING SYSTEM

## ARTICLE REVIEW

DATE: 04/11/2021


**PREPARED BY**

FIONA LE

THANH NGUYEN

Name: Fiona Le, Thanh Nguyen

Paper Review
KungFu: Making Training in Distributed Machine Learning Adaptive

Nowadays, Machine Learning (ML) systems have dealt with highly complex models, and data processing requirements have increased exponentially, making them challenging to provide high performance with a default configuration continuously. Therefore, a new form of ML, namely the Distributed Machine Learning (DML), is becoming a key tool for addressing this problem. DML is implemented by distributing workloads on a cluster of machine nodes, maintaining a globally shared model. Tensor Flow, Horovod, Pytorch are examples of supporting distributed training available for ML developers. These systems support data-parallel training, different partitions of the dataset for each node in the computing cluster, model replication on all worker machines. To utilize the benefit of a DML system, the user must tune multiple parameters to ensure training accuracy and improve training performance. Those parameters have been categorized into two main groups, including hyper-parameters and system parameters. The hyper-parameters, such as the learning rate of workers and input batch size, were tuned to define whether the model is updated and decide the model accuracy. System parameters, such as the total of workers and communication topology, impact the system's training throughput. However, there are existing limitations in using current distributed machine learning (ML) systems that require users to configure parameters manually at a fixed schedule and do not support dynamic parameter updates during training programs.

To resolve the problem, automatic parameter adaptation is initiated by OpenAI. Gradient Noise Scale (GNS) was used to measure variation in data batches and define optimized data batches for achieving good performance. There are many proposals for automatic parameter adaptation recently, such as gradient variance, gradient second-order metrics, worker performance metrics. However, the automatic parameter adaptation is still under some challenges in supporting different adaptation types, adapting based on large volumes of monitoring data, and changing stateful workers' parameters. There is no unified mechanism for adaptation. Custom and external adaptation frameworks are implemented, and these frameworks do not have API to support different types of new adaptation proposals. Training workers dump monitoring data using external libraries, such as TensorBoard, ML flow and perform offline calculation. There is no online way for the system to make real-time adaptation decisions, and those libraries are not designed to train tremendous gradients in real-time. In addition, training workers require the user to checkpoint the state and make a recovery under adaptation to protect the state's consistency. The checkpoint and recovery process, as such, degrade adaptation latency under the number of steps.

The authors have proposed a new distributed ML library, namely KungFu, that enables ML developers to employ adaptive distributed training policies that are able to change configuration parameters automatically and timely during training progress. A new proposed library makes adaptive training more accessible and efficient by designing high-level adaptation policies that implement intuitive API controls. The library also supports flexible combined functions to help ML developers implement different types of adaptations. To ensure the policies are executed during training, embedded monitoring operators are set up during dataflow. New distribution mechanisms for parameter adaptation support online

adaptation toward parameters and allow kungfu workers to expose low-level parameters in a safe manner. Once parameters have been changed, Kungfu can automatically synchronize and update to ensure adaptation is consistent. Comprehensive experiments with large worker stations have proven that KungFu enables practical adaptive distributed training methods and resolves current issues of up-to-date distributed training systems.

The key achievement of Kung Fu is contributed by three components, including Adaptation Policies, Embedding Monitoring Inside Dataflow and Distributed Mechanism for Parameter Adaptation. Instead of manually adjusting parameters at a fixed schedule, Kungfu provided APs created to be flexible in integration with existing ML frameworks, allowing users to pick the right policies for their training scenario and combine multiple policies for more advanced adaptation. The policies will compute metrics to receive a real-time assessment of each worker's performance. The communication functions will be used to combine all local metrics. Based on those metrics, the policies will agree on adaptation decisions for updating parameters to each machine worker in the cluster. Hence, users can modify the policies to meet their expectations, minimize time-consuming and possibly inaccurate induced by external resources for monitoring and adaptation. However, users still need to decide manually on the execution order of all chosen APs. The failure of decisions would impact the whole process and degrade training performance significantly. The mechanism for automatic execution-order-of-APs determination would be the next development step to eliminate the limitation. I suggest authors consider implementing ML to learn the tendency of APs execution order from the human manual decision and then build up a function based on the results to recommend users in choosing the APs' execution order. Kungfu also improves computation resources and spends this resource for model training by efficiently adding monitoring operators into dataflow. This embedded methodology allows the monitoring process to take advantage of a dataflow engine that has been optimized and collective communication operations that can scale out gradients' distributed monitoring. Adopting asynchronous communication between workers using named collective algorithms helps reduce unreliable computation caused by diverse gradients. This method will consume a large amount of memory to store information about states and messages through solving the problem. To resolve this problem, the author would consider utilizing cloud resource sharing across machine workers to minimize internal memory usage. Kungfu provides a distribution mechanism for supporting online parameter adaptation to accelerate execution time and support adaptation progress under low latency. The average value of metrics will be calculated and compiled inside dataflow. This algorithm allows workers to update parameters by recompiling inside the data flow and reuse existing results efficiently.

The authors have performed a detailed benchmark to provide benefits of enabling adaptation in DML and the comparison between Kungfu and existing DML systems in large worker clusters. By implementing adaptation policies for batch size, communication, and resource, Kungfu has demonstrated significant throughput and accuracy compared to baseline. The authors also evaluate the overhead of adaptation and monitoring components between Kungfu and TensorFlow. In the adaptation process, the results show that Kungfu has the processing speed much faster than Tensorflow. The average worker throughput is evaluated in the monitoring process, which comes with and without GNS. The result indicated that the GNS has less impact on throughput, reducing from 6.3% to 1.0% based on

monitoring interval. It illustrates that embedding the monitoring operators as a part of the dataflow graph leads to low overhead. KungFu's distribution performance was compared to Horovod, a most popular DML implementing collective communication technique. The throughput is significantly improved with KungFu implementation than Horovod, in particular with large cluster sizes. In conclusion, Kungfu has provided unique features, addressed multiple issues of existing training distribution systems, and made adaptive MDL easier and more efficiently with minimal user effort.