Evaluating Machine Learning and Big Data Analytics Models on the IHK/McKernel Lightweight Kernel

Student: Sreevatsa Vadiraj
Capstone Advisor: Prof M Mustafa Rafique
Colloquium Advisor: Prof Matthew Fluet
What is a IHK/McKernel

IHK/McKernel is a light-weight multi-kernel operating system designed specifically for High-Performance Computing (HPC) workloads. It runs Linux and McKernel, a lightweight kernel (LWK), side-by-side on the same compute node.
Main Features of IHK/McKernels

• Scalability in executing large-scale applications

• Support existing state-of-the-art programming languages and frameworks with minimal or no changes in the source code

• Nimble support for custom hardware and accelerators

• High-performance due to noise isolation
Motivation

- Tensorflow and PyTorch are widely used in data center environments, and with LWK we aim to speed it up.
- LWKs have proved to have huge performance gains on HPC systems.
- IHK/McKernel powers the world’s fastest super-computer (415.5 petaflops) - Fugaku.
- TF needs to be optimized from the OS point of view.
Solution

• Use IHK/McKernel to run datacenter workloads like Deep learning and Big data analytics models and see if performance is better than a vanilla Linux

• Use horovod to run distributed deep learning models

• Might have to tweak Tensorflow or McKernel to obtain the required gains
Final Deliverable

- A working Tensorflow and PyTorch cluster, running on LWK, giving better performance than vanilla Linux

- Measure only a few workloads related to TF and PyTorch
Results and Evaluation

Success will be measured by showing how different TF and PyTorch workloads and models perform better on the LWK cluster in terms of computation time.
For decades, the lightweight operating system (OS) kernels have been accelerating the performance of various High-Performance Computing (HPC) workloads in supercomputers.

No research has been done to exploit these lightweight kernels (LWKs) in datacenters (cloud computing centers, single, and multi-tenant systems).

Different approaches:
- Two Traditional Approaches
- The Multi-kernel Approach

Another multi-kernel:
- mOS
1. Paper reading

2. Evaluating the performance of applications on a single node and then later expand it to multi-node configuration. Install McKernel on a server

3. On the vanilla Linux, run standard workloads of TensorFlow and PyTorch, such as K-means, Linear Regression, MNIST, and CIFAR image classification

4. Run these same set of workloads on McKernel and record execution time difference with respect to execution on the vanilla Linux
Milestones

5. Create node images of the previous instance of the McKernel running the workloads

6. Create a multi-node setups based on the obtained node images

7. Repeat steps 3 and 4, but with a multi-node configuration for both TensorFlow and PyTorch such that one of the three nodes is the master

8. Report the execution times of the above-mentioned workloads multi-node setup on both vanilla Linux and McKernel
THANK YOU