White Paper

SK Telecom, Intel Build AI Pipeline to Improve Network Quality

Companies build accelerated end-to-end network AI pipelines using Analytics Zoo, TensorFlow, and Apache Spark running on an Intel® architecture server; testing shows solution is six times faster than GPU-based predecessor.[1] The new AI pipeline demonstrates fast predictive analysis of network quality from SK Telecom's huge volume of live datasets and distributed AI application with Spark cluster on 2nd generation Intel® Xeon® Scalable processors, which enables improved network quality prediction for SK Telecom's real-world use cases.

Authors

Hongchan (Nate) Roh
SK Telecom

Jason Dai
Intel

Table of Contents

Introduction 1
AI Pipeline Architecture. 1
Test Parameters 2
Improvements over Conventional Solutions 4
- Improved Model Accuracy 4
- Faster End-to-End Speed 5
- Performance Results 6

Summary 7

Introduction

Network quality is becoming harder for communications service providers (CoSPs) to manage manually because of the overwhelming flood of telemetry data coming from multi-gigabit networks. This challenge grows with the rapid advancement of 5G technology due to the large number of devices and very fast data rates. As a result, managing communication networks in an intelligent and automated fashion using artificial intelligence (AI) technology becomes increasingly important for CoSPs.

SK Telecom, the largest mobile operator in South Korea, manages more than 400,000 cell towers with over 27 million subscribers. This network handles 1.4 million records every second, which accumulates to 120 billion records per day.[2] In order to effectively analyze this massive amount of data, SK Telecom and Intel engineers built an end-to-end network AI pipeline for network quality prediction using Analytics Zoo and FlashBase, running on Intel® architecture servers, which effectively applies a memory-augmented TensorFlow model to large-scale time series data on Apache Spark.

The entire pipeline (from FlashBase to Spark DataFrames to TensorFlow) runs on a unified Intel® Xeon® Scalable processor-based server cluster, with Intel® Advanced Vector Extensions 512 (Intel® AVX-512) and Intel® Deep Learning Boost. Additionally, this leverages Analytics Zoo software to automatically handle the in-memory data pipelines and distributed model training and inferencing. In tests conducted by SKT, this AI pipeline outperforms SKT's legacy GPU-based implementation by up to four times and six times for deep learning training and inference respectively,[1] which enables SK Telecom to more quickly forecast and detect degradation and abnormal changes in network quality so that SKT can take proactive action to deliver their 5G service quality.

AI Pipeline Architecture

Figure 1 shows the high-level architecture for the end-to-end network AI pipeline in use at SK Telecom.

The entire pipeline runs on an Intel Xeon Scalable processor-based Spark cluster using Analytics Zoo. Some of the key elements of the solution include the following:

Analytics Zoo is an Intel-developed open source unified data analytics and AI platform that simplifies scaling a range of AI models for distributed training or inference on a big data cluster (such as Apache Spark). TensorFlow was the

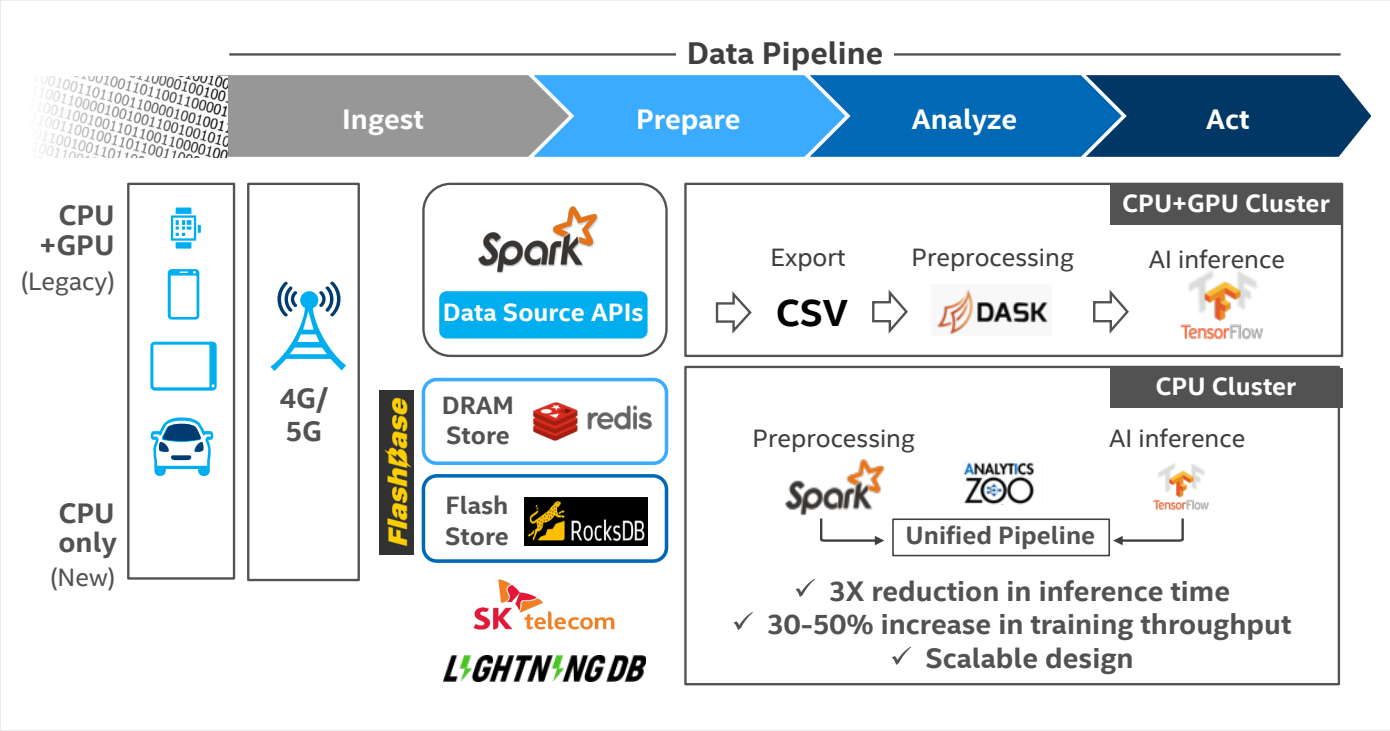


Figure 1. SK Telecom Lightning DB solution using Analytics Zoo on Intel Xeon processor-based servers

AI model used in this testing, but other models are also supported, including PyTorch, Intel® Distribution of OpenVINO toolkit, and Ray. Analytics Zoo also enables porting AI pipelines to Apache YARN or Kubernetes containerized servers without the need to modify the clusters. Analytics Zoo provides unified infrastructure for data processing, model training, and model inference, which reduces data movement and consolidates data storage and pipelines.

Apache Spark is an open-source distributed general-purpose cluster-computing framework. Spark Core provides in-memory computing capabilities to deliver high-performance generalized execution to support a wide variety of applications. APIs for Java, Scala, and Python APIs ensure ease of development. For this testing, DataFrames was used as the programming abstraction language acting as a distributed SQL query engine.

TensorFlow is an open-source symbolic math software library for dataflow and differentiable programming across a range of tasks. TensorFlow is a Python library for fast numerical computing and was created and released by Google. It is a foundation library that can be used to create deep learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

Intel Xeon Scalable processors powered the CPU-only server used in the testing. These CPUs provide the foundation for high performance data center platforms delivering both agility and scalability. This innovative processor platform converges capabilities across compute, storage, memory, network, and security. The Intel Xeon Scalable platform is designed for data center modernization to drive operational efficiencies that lead to improved total cost of ownership (TCO) and higher productivity for users.[3] For AI projects, Intel Xeon Scalable processors provide vectorized and deep learning instructions via support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) and Intel® Deep Learning Boost.

Test Parameters

Network KPI data is collected every five minutes from over 400,000 cell towers and stored into FlashBase, an in-memory data store for Spark that supports extreme data partitioning and efficient aggregation push-down. The KPI data is then processed using Spark DataFrames. After that, Analytics Zoo can directly apply the TensorFlow models to the in-memory Spark DataFrames in a distributed fashion across the Spark cluster, as illustrated below.

1. First, load the data from FlashBase using Spark DataFrames for preprocessing, and transform Spark DataFrames to the resilient distributed datasets (RDD) of TensorFlow Tensors.

```
from zoo.tfpark import TFDataset
train_df, val_df = get_df_from_flashbase(sc, train_
cells=0.8, valid_cells=0.2)
dataset = TFDataset.from_dataframe(train_df,
                feature_cols=[...],
                  labels_cols=[...],
                  batch_size=config.batch_size,
                  val_df=val_df)
```

2. Second, use standard TensorFlow APIs to build the memory-augmented network model.

```
class Model(object):
  def __init__(self, config, input_x=None, memories=None, targets=None):
    ...
    self._build_model()

  def _build_model(self):
    self.add_placeholder()
    with tf.variable_scope("inputs"):
      input_ar, ar_loss = self.auto_regressive(self.input_x, ...)
      with tf.variable_scope("memories"):
        memories = tf.concat(tf.split(self.memories, ...), axis=0)
        memory_ar, ar_loss_ = self.auto_regressive(memories, ...)
        context = self.attention(input_ar, memory_ar)
        linear_inputs = tf.concat([input_ar, context], axis=1)
        self.predictions = tf.layers.dense(linear_inputs, ...)
        self.loss = tf.losses.mean_squared_error(labels=self.targets,
                                                 predictions=self.predictions)

model = Model(config, dataset.feature_tensors[0], dataset.feature_tensors[1],
          dataset.label_tensors)
```

3. Third, use Analytics Zoo APIs to train the TensorFlow model on the Spark cluster in a distributed fashion at scale.

```
optimizer = TFOptimizer.from_loss(model.loss, Adam(1e-3))
optimizer.optimize(end_trigger=MaxEpoch(num_epochs))
saver = tf.train.Saver()
saver.save(optimizer.sess, "/tmp/armem")
```

4. Finally, use Analytics Zoo APIs for distributed TensorFlow inference at scale on Spark DataFrames.

```
dataset = TFDataset.from_dataframe(test_df, feature_cols=[...], batch_per_thread=4)
model = Model(config, dataset.feature_tensors[0], dataset.feature_tensors[1])
sess = tf.Session()
saver = tf.train.Saver()
saver.restore(sess, "/tmp/armem")
predictor = TFPredictor.from_outputs(sess, [model.predictions])
predictions_rdd = predictor.predict()
```

Improvements over Conventional Solutions

Several innovations were adopted in this end-to-end network AI pipeline in order to improve both model accuracy and end-to-end performance over the conventional GPU-based architecture that SKT had previously utilized.

Improved Model Accuracy

Seq2seq is an encoder/decoder framework and is often used for sequence prediction. However, this turns out to be a suboptimal implementation for this type of network AI use case, primarily because the Seq2seq model cannot predict the sudden changes in the network KPI data (as shown in Figure 2).

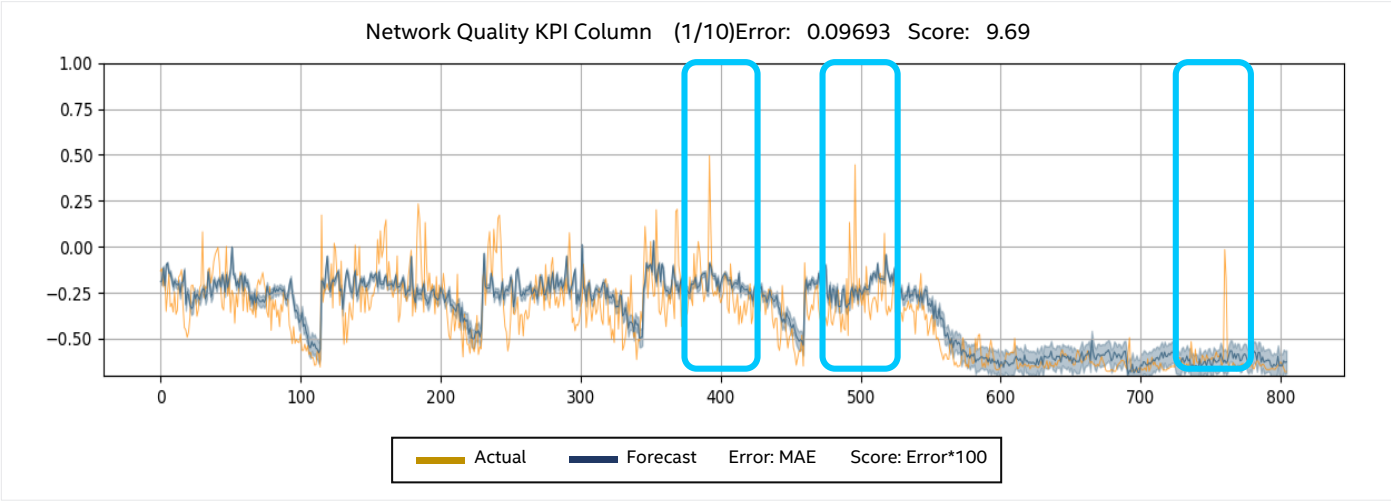


Figure 2. Seq2seq cannot predict sudden changes in network KPIs

Based on work done by Chang et al.,[4] the authors built a new memory-augmented model to improve the prediction accuracy of network KPIs, as shown in Figure 3.

- The model takes two inputs, namely the data collected in the last 50 minutes, and the history data of the last 7 days during the same time period. It then predicts the future value for the next 5 minutes.
- Data in the last 50 minutes and last 7 days are passed to two different encoders that are defined by autoregressive terms.
  - Encoder 1 : $h_t = c + w_1 y_{t-ndays-1} + \ldots + w_{11} y_{t-n\,days-11}$
  - Encoder 2 : $h'_t = c' + w'_1 y_{t-1} + \ldots + w'_{10} y_{t-10}$
- In the next step, attention scores are calculated, and the attention weighted memories are concatenated with the current state vector for final prediction.

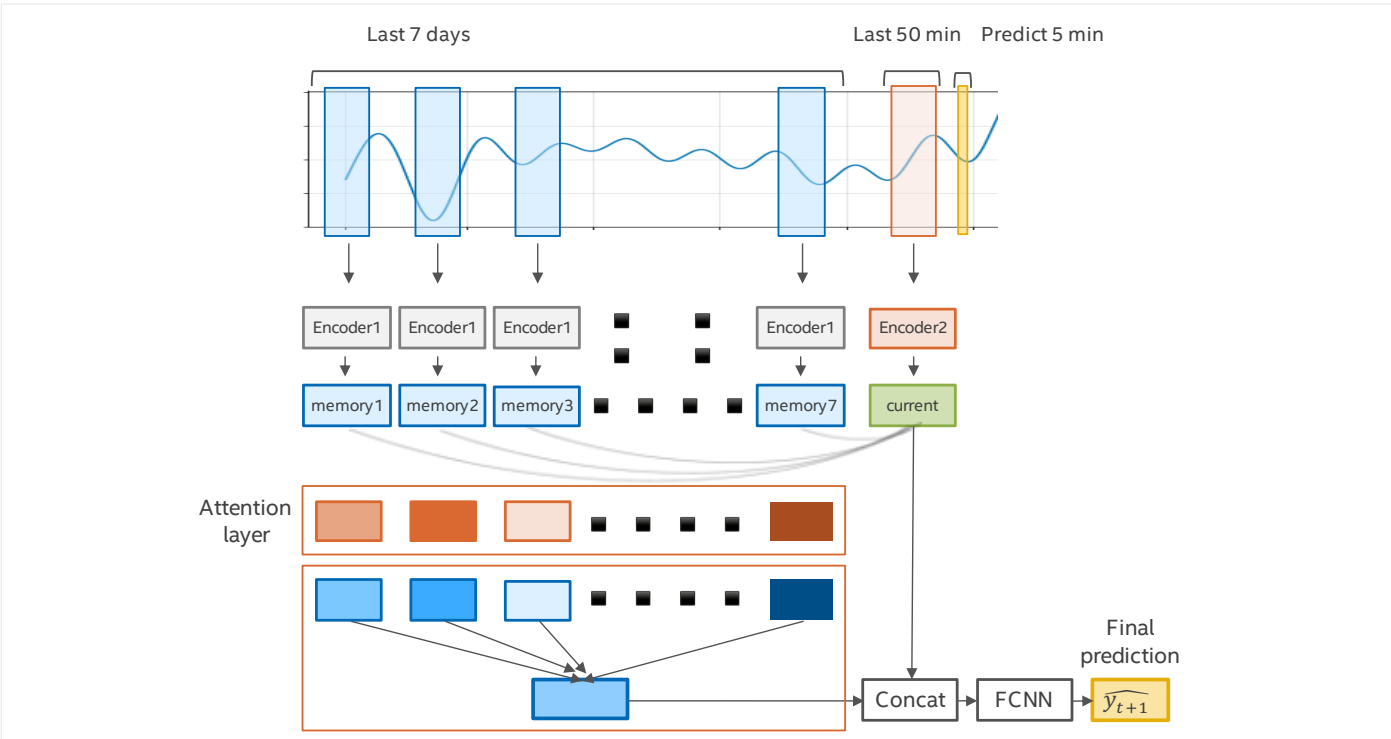


Figure 3. Memory-augmented network model

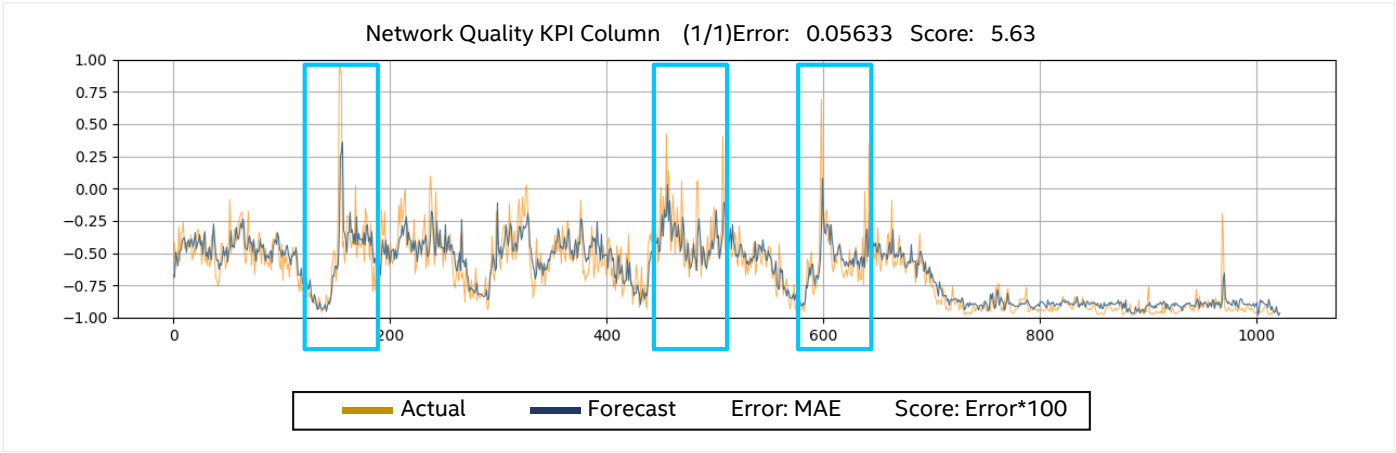


Figure 4. Memory-augmented model can predict sudden changes

As shown in Figure 4, the memory-augmented model can accurately predict any sudden changes in network KPI data, which is critical in the network quality prediction use case.

Faster End-to-End Speed

Previously, SK Telecom followed a conventional approach to set up two separate clusters, one for data processing in Spark, and the other for deep learning training/inference using GPUs, as illustrated in Figure 5. However, this created two separate workflows, which introduced significant overhead, including exporting data from Spark cluster through files, copying the files between different clusters, and loading the files from disks—all of which caused delays and increases the maintenance burden.

By moving to the new CPU-based architecture shown in Figure 1, SKT could run the end-to-end pipeline on the same Spark cluster using Analytics Zoo in a distributed fashion to unite the data store (using FlashBase), data preprocessing (with Spark DataFrames), model training and inference (with TensorFlow) into an integrated in-memory data analytics pipeline. This integrated pipeline performance was four times faster for deep learning training and six times faster for inference than the existing GPU architecture.[1]

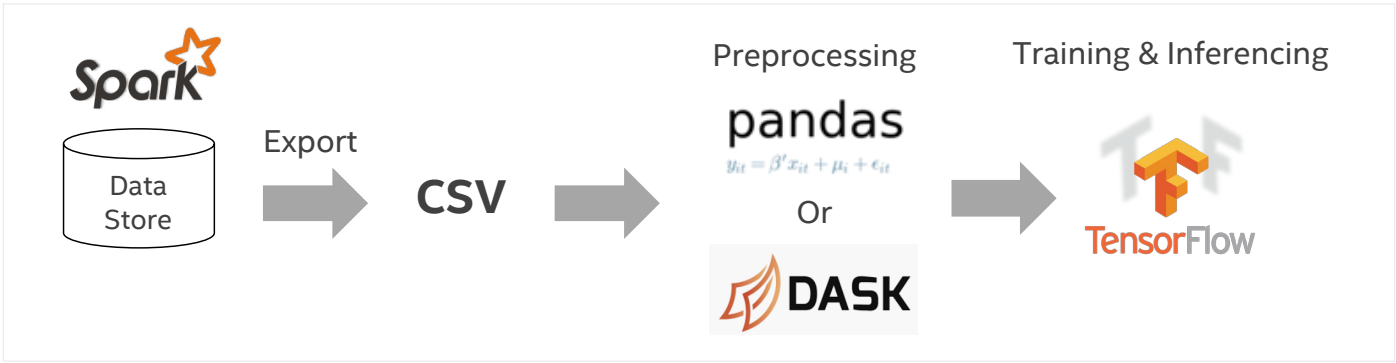
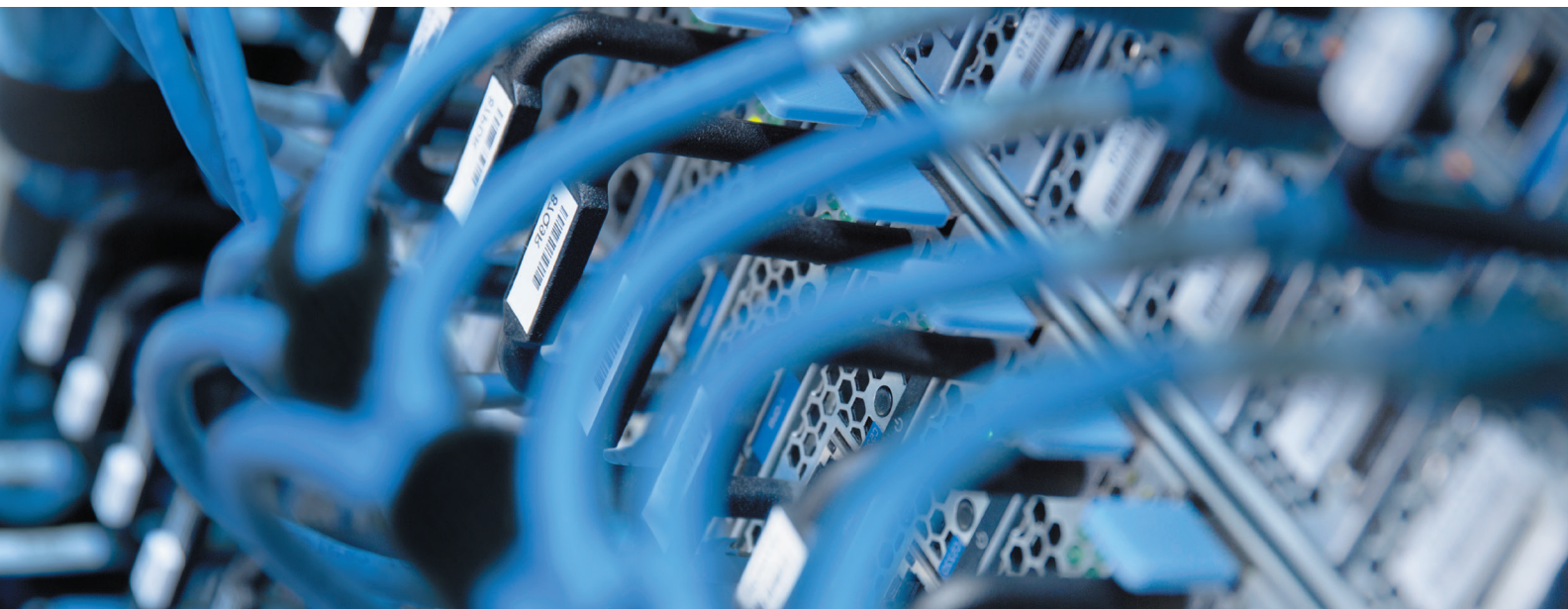


Figure 5. Old architecture with two separate clusters



Performance Results

Figure 6 shows the training throughput (record/second) for Analytics Zoo running on Intel Xeon Gold 6240 CPU-based server cluster compared to the legacy architecture based on NVIDIA GPU . As shown in Figure 7, Analytics Zoo running on a single-node Intel Xeon Gold 6240 server demonstrated competitive training performance compared with one GPU. In addition, the Analytics Zoo solution can seamlessly and efficiently scale to a large cluster using Spark, which showed up to 4x speed-up on a 3-node Intel Xeon Scalable cluster compared to one GPU.

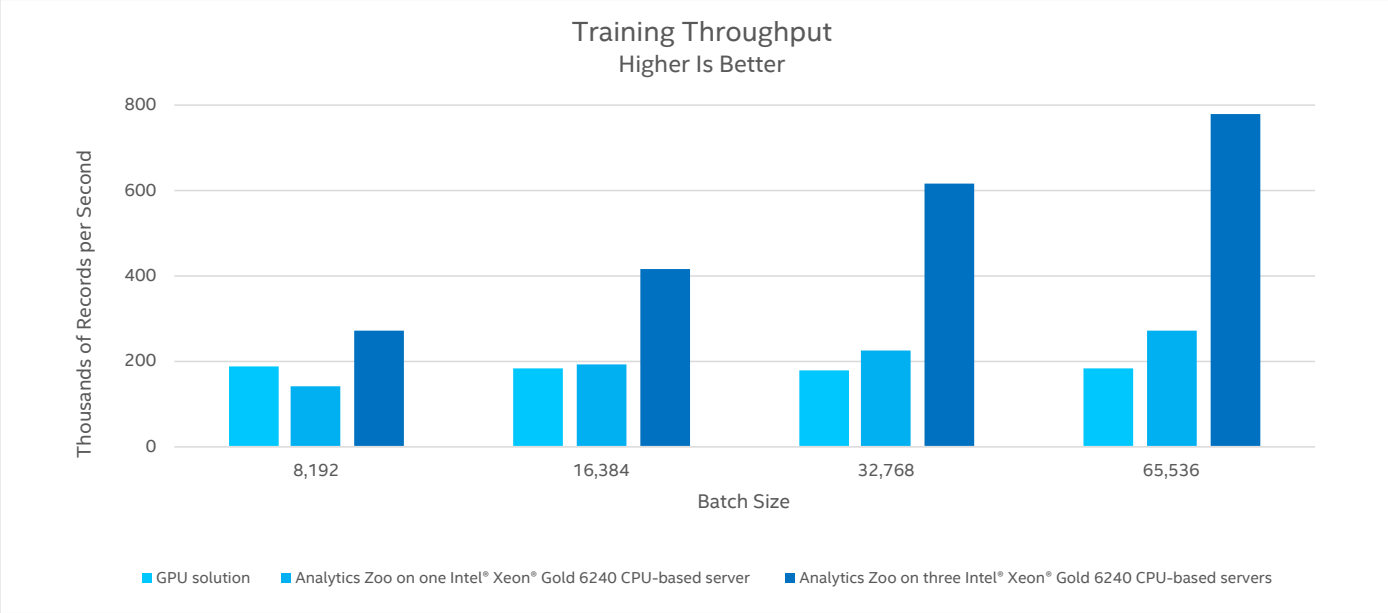


Figure 6. Training throughput tested at SK Telecom Testbed (higher is better)

Figure 7 shows the performance in terms of the elapsed time for the end-to-end inference pipeline, which includes data load, data preprocessing, and model inference. (Note that data copy overhead to the GPU architecture is not included here). As shown in Figure 7, Analytics Zoo running on Intel Xeon Scalable Gold 6240 processor-based servers outperformed the legacy GPU-based solution by up to three times on a single-node server, and up to six times when running on a three-node cluster for preprocessing and inference stages.

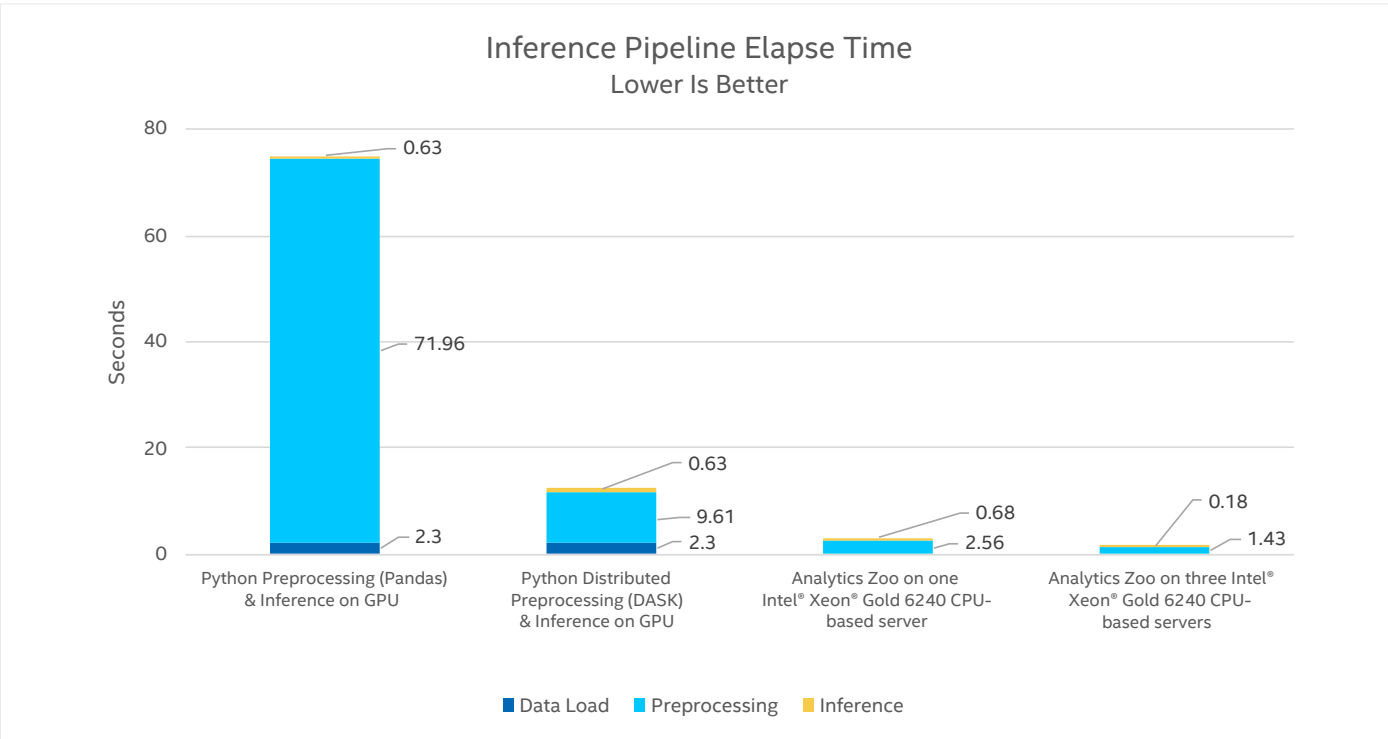


Figure 7. Elapsed time for end-to-end inference pipeline tested at SK Telecom Testbed (lower is better)

Summary

SK Telecom have developed an end-to-end network AI pipeline using open source FlashBase, Spark, TensorFlow, and Analytics Zoo. Leveraging an Intel® Xeon® Scalable processor-based server cluster, with Intel® Advanced Vector Extensions 512 (Intel® AVX-512) and Intel® Deep Learning Boost, SK Telecom can execute unified network quality prediction workloads, including data processing to feature engineering and deep learning training/inference, as an integrated in-memory data analytics pipeline. In tests conducted by the company, this pipeline architecture outperformed SKT's legacy GPU-based pipeline architecture by up to six times.

Learn More

Intel Xeon Processors: https://www.intel.com/xeon

SK Telecom: https://www.sktelecom.com/index_en.html

Analytics Zoo: https://software.intel.com/content/www/us/en/develop/topics/ai/analytics-zoo.html

Apache Spark: https://spark.apache.org

TensorFlow: https://www.tensorflow.org

Lightning DB: https://lightningdb.io

About the Authors

Hongchan (Nate) Roh
hongchan.roh@sk.com
Team leader (Sr. Software Engineer), SK Telecom

DBMS Ph. D., Data engineer, and creator of Lightning DB (real-time big data analytics engine)

Jason Dai
jason.dai@intel.com
Sr. Principal Engineer, Intel, Intel Architecture Graphics and Software

Founding committer and PMC member of Apache Spark; mentor of Apache MXNet; creator of BigDL and Analytics Zoo

Notices & Disclaimers

[1] Tests conducted by SK Telecom in Feb. 2020: The Analytics Zoo server was an Intel® Server System R2208WFTZSR powered by a 2.6 GHz Intel Xeon Gold 6240 processor (microcode 0x400002c). The server featured three nodes and six sockets. Both Intel® Hyper-Threading Technology and Intel® Turbo Boost Technology were turned on. Total memory equaled 256 GB. CentOS 7.8 (kernel 3.10.0) was the operating system and the server ran the SK Telecom Lightning DB application. Other software included Analytics Zoo v0.7, Tensorflow v1.15, Pandas v0.25.3, NumPy v1.18.0, and Dask v2.7.0.

The GPU server was a HPE DL380 Gen 9 powered by a 2.4 GHz Intel Xeon E5-2680 v4 processor (microcode 0xb00001e) and an Nvidia P100 GPU (AI training)/K80 (AI inference). The server featured one node and two sockets. Both Intel Hyper-Threading Technology and Intel Turbo Boost Technology were turned on. Total memory equaled 256 GB. CentOS 7.3 (kernel 3.10.0) was the operating system and the server ran the SK Telecom Lightning DB application. Other software included Tensorflow GPU v1.12, Pandas v0.25.1, NumPy v1.14.5, and Dask v2.7.0.

[2] Data from SK Telecom, September 2020.

[3] https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/xeon-scalable-platform-brief.pdf

[4] Yen-Yu Chang, Fan-Yun Sun, Yueh-Hua Wu, Shou-De Lin. "A Memory-Network Based Solution for Multivariate Time-Series Forecasting". https://arxiv.org/abs/1809.02105

1020/DO/H09/PDF ♻ Please Recycle 344751-001US