Solution Brief

Cross-Industry Architecture

Intel® Optimized Cloud Modules

intel software

Intel® Optimized Cloud Modules for Microsoft Azure*: XGBoost* Pipeline on Kubernetes*

Author: Kelli Belcher

September 05, 2023

Deploy scalable, accelerated, and secure machine learning pipelines on an Azure confidential cluster

The **Intel® Optimized Cloud Modules for Microsoft Azure*: XGBoost* Pipeline on Kubernetes*** is designed to facilitate building and deploying highly available and scalable AI applications on **Microsoft Azure**. This reference solution provides an optimized training and inference pipeline using XGBoost to predict the probability of a loan default. The module enables the use of **Intel® Optimization for XGBoost***, **Intel® oneAPI Data Analytics Library (oneDAL)**, and **Intel® Extension for Scikit-Learn*** in a full end-to-end reference architecture. It also leverages secure and confidential computing using the **Intel® Software Guard Extensions (Intel® SGX)** virtual machines on an **Azure Kubernetes Services (AKS)** cluster.

Use it as a reference solution for:

- Data preprocessing
- Machine learning for binary classification tasks
- Model inference and performance analytics
- Applications deployed on Kubernetes clusters
- Confidential computing services

Who needs it?

- Developers aiming to maximize performance and productivity with industry-leading, CPU-accelerated Python* libraries for machine learning
- Developers with access to new or existing Azure Kubernetes applications who want to take advantage of secure and confidential computing virtual machines leveraging Intel® Software Guard Extensions

What it does

This module demonstrates how to build a scalable, accelerated machine learning pipeline on a secure and confidential computing cluster. The module serves as a guide for the following:

- Configuring Microsoft Azure cloud services, including an **Azure resource group**, an **Azure file share**, an **Azure load balancer**, an **Azure container registry (ACR)**, and an **Azure Kubernetes Services (AKS)** cluster with a confidential computing node pool.
- Implementing a full, end-to-end machine learning pipeline to predict credit risk using an XGBoost model, from data preprocessing to model inference with Intel oneAPI software optimizations.
- Leveraging the security of confidential computing nodes on an Azure Kubernetes Services cluster using Intel® SGX virtual machines powered by 3rd Generation **Intel® Xeon® Scalable processors,** and using **Intel® Turbo Boost Max Technology 3.0** to reach 3.5 GHz for AI acceleration.

Cloud Solution Architecture

This solution architecture uses **Docker*** for application containerization and stores the image in an Azure container registry. The application is then deployed on a cluster managed by Azure Kubernetes Services. Our cluster runs on confidential computing virtual machines leveraging the enhanced security provided by Intel® Software Guard Extensions. The application utilizes a mounted Azure file share for persistent data and model storage. An Azure load balancer is provisioned by our Kubernetes service that the client uses to interact with the application. These Azure resources are managed in an Azure resource group. When the application is deployed, each pod will be assigned to an Intel SGX node. Figure 1 shows the cloud solution architecture.

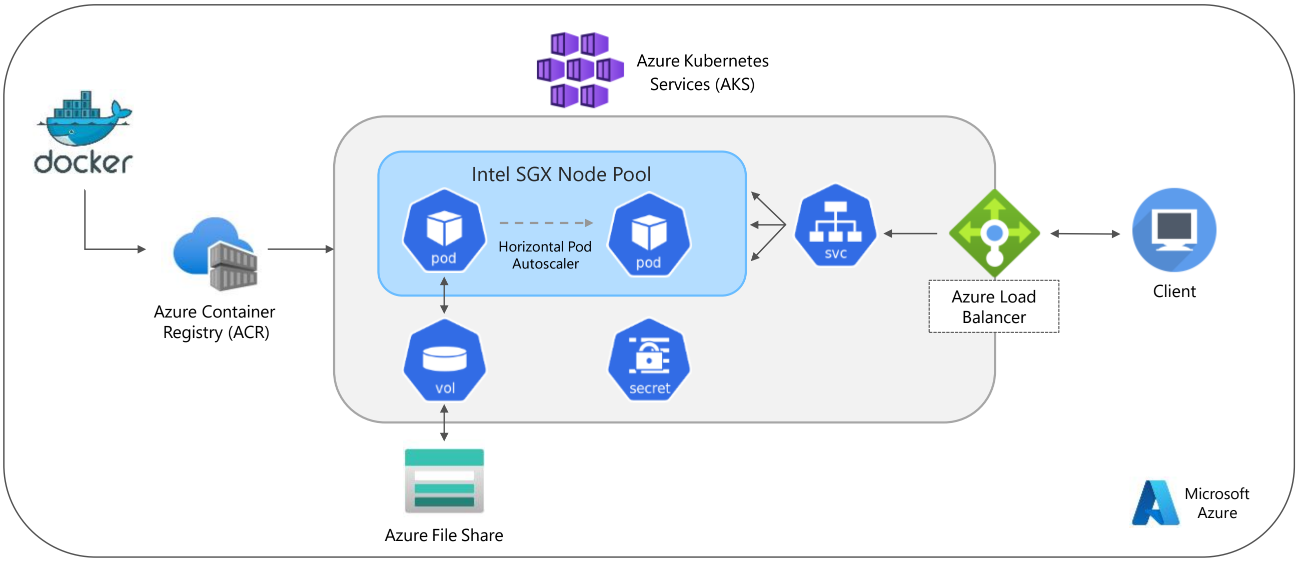

Figure 1: XGBoost Pipeline on Kubernetes Architecture Diagram. Image from author.

Solution Component Overview

This solution is derived from the **Loan Default Risk Prediction AI Reference Kit**, built by Intel in collaboration with Accenture. The code has been enhanced through refactoring to achieve better modularity and suitability in support of our three main APIs: data processing, model training, and inference. The credit risk data set used in the application is obtained from **Kaggle*** and synthetically augmented for testing and benchmarking purposes.

The XGBoost Pipeline on Kubernetes consists of the following three main API endpoints:

1. **Data processing**
   This endpoint receives the original credit risk CSV file, creates a data preprocessing pipeline, and generates a training and test set of the size specified. The train and test data, as well as our preprocessor, are stored in the Azure file share.
2. **Model training**
   This endpoint begins training an XGBoost classifier. The model and the validation results are stored in the Azure file share. Additionally, this endpoint provides an option to continue training the model as new data becomes available.
3. **Inference**
   This endpoint retrieves the trained XGBoost model from the Azure file share and converts it into an inference-optimized daal4py model to calculate the predictions on test data. The inference results are then returned to the client and stored in the file share, containing the predicted risk probability and a corresponding label of True when the probability is greater than 0.5, or False if otherwise.

Code Highlights

Enable the Intel Optimizations for XGBoost

The **XGBoost optimizations** for training and inference on CPUs are upstreamed into the open source XGBoost framework. Ensure you are using the latest version of XGBoost to access the most Intel optimizations. No changes are needed to the underlying code. The following code demonstrates training of the classification model using XGBoost version 1.7.6.

```
# Create the XGBoost DMatrix, which is optimized for memory efficiency and training speed
dtrain = xgb.DMatrix(X_train.values, y_train.values)

# Define model parameters
params = {"objective": "binary:logistic",
     "eval_metric": "logloss",
     "nthread": 4,  # num_cpu
     "tree_method": "hist",
     "learning_rate": 0.02,
     "max_depth": 10,
     "min_child_weight": 6,
     "n_jobs": 4,  # num_cpu,
     "verbosity": 1}

# Train initial XGBoost model
clf = xgb.train(params = params, dtrain = dtrain, num_boost_round = 500)
```

Convert the Trained XGBoost Model to daal4py

daal4py is the Python API of the oneAPI Data Analytics Library, oneDAL. daal4py helps to further optimize model prediction, or inference, on CPUs. The following code demonstrates how to convert a trained XGBoost model into daal4py format and calculate the predicted classification results.

```
# Convert XGBoost model to daal4py
d4p_model = d4p.mb.convert_model(clf)

# For optimized inference on CPU
d4p_probabilities = d4p_model.predict_proba(data)
```

Enable Intel Extension for Scikit-Learn

Intel Extension for Scikit-Learn* provides CPU accelerations for many scikit-learn libraries. Below is an example using the scikit-learn extension to accelerate the computation of the Area Under the ROC Curve (AUC).

```
# Call patch_sklearn() before importing scikit-learn libraries
from sklearnex import patch_sklearn
patch_sklearn()

from sklearn.metrics import roc_auc_score

# Use the CPU-accelerated version to calculate the Area Under the ROC Curve
auc = roc_auc_score(y_test, d4p_probabilities)
```

Next Steps

Download the module from GitHub ›

Register for office hours to get help on your implementation ›

Check out the full suite of Intel® Optimized Cloud Modules ›

Come chat with us on our DevHub Discord* server to keep interacting with other developers ›

Intel® technologies may require enabled hardware, software, or service activation. Learn more at intel.com or from the OEM or retailer. Your costs and results may vary.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Optimization notice: Intel® compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel® microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel® microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product user and reference guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804. https://software.intel.com/en-us/articles/optimization-notice

Software and workloads used in performance tests may have been optimized for performance only on Intel® microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. See backup for configuration details. For more complete information about performance and benchmark results, visit intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and noninfringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

1121/SS/CMD/PDF