Intelligent Data Processing System

Motivation

Recently, GPU-accelerated systems, such as TensorFlow, PyTorch, and Caffe, are increasingly studied and deployed in the big data world. To deploy trained deep learning (DL) models for inference on the large-scale dataset, the intelligent data processing systems (IDPS) usually consist of two parts [1][2]: an analytics system and an inference system. From a functional perspective, current vanilla IDPS solutions, such as TensorFlowOnSpark [1], have provided an easy way to deploy inference applications at scale. However, they face challenges in the data-intensive and time-sensitive scenarios. Considering typical cases, such as image or video analysis in smart cities and financial fraud detection, massive input data should be processed within a limited time. Otherwise, the timeliness of inference results will degrade inevitably [3], when the underlying execution is suboptimal due to the separated data path of analytics and inference.

We realize a subtle fact that even though they belong to different domains, both of them are data-intensive, where a large volume of data flows through CPU or GPU. As a result, we optimize the IDPS workloads by being aware of both analytics and inference, rather than just assembling them in a vanilla manner, which also inspires similar solutions with other hardware accelerators like FPGA/RISC-V accelerators.

Background

To deploy an end-to-end intelligent data processing application, the inference system will work with analytics systems. To cope with massive data at scale, communities have contributed various systems (e.g., Spark) along with their extensions. In these systems, application data can be abstracted as relational tables. Apart from common scalar data types like integer and string, the binary type is used in inference workloads, such as image or video analysis. Then an application is organized by a series of operations on the dataset. For the most common map operation, it intakes the input dataset D to produce a new dataset D' by applying a user-defined lambda function λ (i.e., $e \to \lambda(e)$) on each item e of D. Analytical operations, such as projection and filter, can be filled into λ as its payload to achieve data pre-processing on CPU. To cope with DL inference, the pre-trained DL model m is applied in the form of lambda function λ_m. Unlike typical analytical operations, λ_m is a series of inference kernels along with the weights of the model m.

With IDPS, the analytics part carries out general-purpose, lightweight, and CPU-based operations, such as input pre-processing and result aggregation. When it encounters DL inference processing, a batch of items will be delivered to the inference part that carries out GPU execution for the given DL model. There are a series of challenges and opportunities when we integrate the two kinds of systems to build a new kind of intelligent data processing system, and implement intelligent inference on the large-scale dataset by changing the fundamental architecture, for example, how to leverage new hardwares such as NVM/GPU/FPGA to accelerate the intelligent data processing system, how to reduce the overhead of PCIe transfer between CPU and GPU sides, and how to deploy it in the cloud along with the local cluster.

Methodology

My approach combines theoretical analysis techniques and fundamental systems insights. Both are crucial: theoretical analysis providing strength over ad hoc approaches, and giving my research a considerable depth; constructing the working tools and prototypes, and applying them to validate the proposed methods in real systems.

With the methodology, I have conducted research on the intelligent data processing system. To eliminate cache misses in SIMD vectorization, we presented interleaved multi-vectorizing (IMV) (VLDB2020), which interleaves multiple execution instances of vectorized code to hide memory access latency with more computation. IMV could make full use of the data parallelism in SIMD and the memory level parallelism through prefetching. To accelerate data processing with GPU, we presented XeFlow (TC2020, IEEE Transactions on Computers) that enables streamlined execution by leveraging hardware mechanisms inside new generation GPUs. XeFlow significantly reduces costly explicit copy and kernel launching within existing CKC or its variants. To accelerate data analytics with GPU, we decoupled the control plane and the data plane within big data systems via action shadowing. The control plane keeps logic information to fit well with the host systems like Spark, while the data plane holds data and performs execution upon bare metal CPUs and GPUs. We implemented an accelerated data plane, namely ShadowVM (PPoPP2021), which significantly outperformed the JVM-based Spark, and the GPU-only fashion by adopting mixed CPU-GPU execution. To fully exploit the hardware potential of NVMe devices, we proposed a lightweight native storage stack called Lightstack to minimize the software overhead. The core of Lightstack is an efficient table storage engine, LATTE (ICDE2020), which abstracts the essential data service of the database's 2D table.

In addition, to improve the performance of big data applications running in the cloud platform, we proposed a hybrid scheduling framework and strategy for scheduling virtual CPUs on the virtualized multi-core systems, and further proposed adaptive scheduling framework for virtual CPU assignment. Some parts of the work were published at IEEE Transactions on Computers, HPDC2011 and VEE2009, which were followed by papers at EuroSys2011, ASPLOS2013, etc. We analyzed the potential variety of attacks in virtual machines in the cloud platform, and established a multi-level access control model for enforcing isolation in virtualization. Some parts of the work were published at TC2015, TC2016 and TDSC2021 (IEEE Transactions on Dependable and Secure Computing).

Besides, we developed Ginkgo (https://github.com/daseECNU/Ginkgo), a working distributed data management and processing system for data processing applications. Further, we proposed a dynamic scheduling algorithm, List with Filling and Preemption (LFPS) (TKDE2020, IEEE Transactions on Knowledge and Data Engineering), to address the issue of scheduling resources to multiple pipelines of one query in a main memory database cluster. Based on proposed metadata-oriented protocol, Karst (TKDE2020) converts a distributed transaction into multiple partial transactions to avoid the two-phase commit, which also employs lazy persistence, lightweight logging, and optimized data traffic.

Based on my research experiences and preliminary efforts, in my future research, I will focus on the intelligent data processing systems from the aspect of interdisciplinary systems, intersecting big data systems, deep learning systems, and hardware and emerging technologies. Specifically, (1) Through leveraging GPU/FPGA/SIMD/In-Memory technologies, I will conduct research on accelerating the intelligent data processing systems with new models and algorithms. (2) The light-weight storage is a feasible mechanism for leveraging the emerging memory technologies, so I will study how to establish the new storage software stack to achieve a significant performance improvement for the intelligent data processing systems. (3) I will also study system security and efficiency at the same time when we deploy those applications in the cloud, and attempt to establish effective-security methods to improve their security while keeping their performance.

Reference

[1] 2021. TensorFlowOnSpark. https://github.com/yahoo/TensorFlowOnSpark.

[2] 2021. Spark DL Inference Workflow. https://docs.databricks.com/applications/machine-learning/model-inference/index.html

[3] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. 2019. Nexus: A GPU Cluster Engine for Accelerating DNN-based Video Analysis. In Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP). 322–337.