Performance-Efficient E2E AI Pipelines

Meena Arunachalam, Principal Engineer, Artificial Intelligence and Analytics Group
Intel Corporation

13th International Workshop on Accelerating Analytics and Data Management Systems Using Modern Processor and Storage Architectures
In conjunction with VLDB 2022 ADMS Workshop 2022
Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.
No product or component can be absolutely secure
Your costs and results may vary.
© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.
Agenda

- Accelerated growth of AI demands on compute/memory/system
- AI HW and SW acceleration
- Multi-phased End-to-End AI optimization strategy
  - Data+AI SW, frameworks, graph compilers and low-level libraries
  - Low precision quantization
  - Learning efficiencies
  - System level optimizations
  - Hyperparameter optimizations
- Performance-efficient E2E AI pipelines
- Call to action
Data + Model + Deploy
AI Loves TFLOPS?

But ... how do we feed all that compute?

A. Gholami, et al., *AI and Memory Wall*. RiseLab Medium Blog Post, 2021
AI and the Memory wall

Transformer Size: 240x / 2 yrs
AI HW Memory: 2x / 2 yrs

A. Gholami, et al., AI and Memory Wall. RiseLab Medium Blog Post, 2021
AI Hardware Acceleration

GENERAL PURPOSE

CPU

GPU

PURPOSE BUILT

ACCELERATORS
Does Software Acceleration Matter?

Up to 10 - 100x with SOFTWARE Acceleration

Hardware Acceleration
End-to-End AI Software Suite
End-to-End AI Optimization strategies

- Model Optimizations
  - Quantization (BF16, INT8)
  - Pruning
  - Knowledge distillation

- Data + AI SW Acceleration

- Learning Optimizations
  - Transfer learning
  - Fine tuning
  - Few shot learning
  - Active learning
  - Filtration and others

- Performant & Efficient AI

- System Level Tuning
  - CPU Turbo mode
  - Hyperthreading
  - CPU power scaling governor
  - Sub-NUMA clustering
  - Transparent Huge Pages
  - Hardware prefachers, memory interleaving

- Workload Scaling
  - Core scaling
  - Instance scaling
  - Node scaling
  - Improve load balancing, reduce serial code

- Parameter Optimizations
  - OMP num threads, KMP affinity etc.
  - HPO: Batch size, learning rate, max depth, L1/L2 normalization, etc. using AutoML based tools [SigOpt]
Deep Learning powered by oneDNN

A simple program is good, but may be slow

Optimizations: vectorization, data reuse, parallelization

Optimized convolution in oneDNN
Graph Optimization Example

Baseline

BN Folding

Conv + ReLU

Conv + Sum

INT8 Optimized Model

\[
\begin{align*}
A_0 & \quad A_1 & \quad A_2 & \quad A_3 & \quad \ldots & \quad A_{63} \\
B_0 & \quad B_1 & \quad B_2 & \quad B_3 & \quad \ldots & \quad B_{63} \\
C_0 & \quad \ldots & \quad \ldots & \quad \ldots & \quad C_{15} \\
A_0 B_0 + A_1 B_1 + A_2 B_2 + A_3 B_3 + A_4 B_4 + A_5 B_5 + A_6 B_6 + C_0 & \quad \ldots & \quad A_{60} B_{60} + A_{61} B_{61} + A_{62} B_{62} + A_{63} B_{63} + C_{016}
\end{align*}
\]
Machine learning powered by oneDAL

1. The best performance on Intel Architectures with oneMKL (Intel® MKL) vs. lower performance on BLAS/LAPACK libs

2. oneDAL targets to many-core systems to achieve the best scalability on Intel® Xeon, other libs mostly target to client versions with small amount of cores

3. oneDAL uses the latest available vector-instructions on each architecture, enables them by compiler options, intrinsic. Usually, other ML libs build applications without vector-instructions support or sse4.2 only.

4. oneDAL uses the most efficient memory optimization practices: minimally access memory, cache access optimizations, SW memory prefetching. Usually Other ML libs don’t make low-level optimizations.

5. oneDAL enables new instruction sets and other HW features even before official HW launch. Usually, other ML libs do this with long delay.

6. oneDAL provides distributed algorithms which scale on many nodes
Low precision Inference: Model quantization

Model quantization to low precision yields significant performance speedups on a variety of models. Pruning, distillation and mixed precision strategies also further model efficiencies.
Hyper parameter and Instance tuning with SigOpt (DIEN)

- https://sigopt.com/
- Sigopt features:
  - Easy to track runs, visualize training, and scale hyperparameter optimization
  - Advanced optimization Engine that delivers better results, faster and cheaper
  - Easy to use and parallelize for any type of model built with any library on any infrastructure
System level Tuning

Intel Xeon Platinum 8380 Processor with 40 cores per socket

Hyperthreading allows 2 threads to run on a core

Performance scaling drivers control core frequencies and power configurations

Examples of BIOS/system level tuning:
- Hyperthreading
- CPU Turbo Boost Technology
- CPU power scaling governors
- NUMA optimizations
- Sub-NUMA Clustering
- Transparent Huge Pages
- Hardware prefetchers
- Channel interleaving
- Memory interleaving
- Hardware P-state
- Hardware C-state

Sub-NUMA Clustering divides the cores, cache, and memory of the processor into multiple NUMA domains and helps workloads that are NUMA-aware be optimized.

Channel interleaving divides memory blocks and spreads contiguous portions of data across interleaved channels, thereby increasing potential read bandwidth.

Memory Interleaving allows physical ranks of memory to be accessed while another is being refreshed.

Instance scaling: This example is 10 Xeon instances per socket
PLAsTiCC Astronomical Classification

PLAsTiCC is an open data challenge to classify objects in the sky that vary in brightness using simulated astronomical time-series data. The challenge is to determine a probability that each object belongs to one of 14 classes of astronomical filters.
**End-to-end ML optimizations**

**PLAsTiCC Astronomical Classification**

<table>
<thead>
<tr>
<th>Ingestion</th>
<th>Feature Engg</th>
<th>Machine Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Read data</td>
<td>• Drop columns</td>
<td>• Model prediction</td>
</tr>
<tr>
<td>• Create dataframe</td>
<td>• Groupby agg</td>
<td>• Calculate accuracy</td>
</tr>
<tr>
<td></td>
<td>• Arithmetic ops</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Create feature set/test set</td>
<td></td>
</tr>
</tbody>
</table>

**Modin**

Modin transparently distributes the data and computation across available cores, unlike Pandas which only uses one core at a time.

Modin can be installed from PyPI:

```
pip install modin
```

# import pandas as pd
import modin.pandas as pd

Single line import change to run Modin instead of pandas.

**Scikit-learn**

Foundational library to speed up your Scikit-learn application, that is highly optimized with low-level HW feature enabling to cover data analytics and machine learning.

```
from sklearn.svm import SVC
X, Y = get_dataset()
clf = SVC().fit(X, y)
res = clf.predict(X)
```

Scikit-learn mainline

**XGBoost**

Intel’s optimizations are now available as part of mainline XGBoost repository.

```
from sklearn.svm import SVC
X, Y = get_dataset()
clf = SVC().fit(X, y)
res = clf.predict(X)
```

Available through PyPi

```
pip install scikit-learn-intelex
```
PLAsTiCC Astronomical Classification

Performance with optimized software & hyperparameters

- **14GB dataset with 1.4 millions rows in training and 189 million rows in test dataset** – takes advantage of Modin’s extremely light-weight, robust Dataframe & readcsv operation which scales with cores, unlike pandas

- Operations in feature engineering are memory bound and benefit from the faster memory access speeds

- **Microarchitecture factors like higher core frequency, cache size, cache BW help improve XGBoost performance**

---

Hardware: 2 x Intel Xeon Platinum 8280L (28 cores), OS: Ubuntu 20.04.1 LTS Mitigated, 384 GB RAM (384 GB RAM: 12 x 32 GB 2933 MHz), kernel: 5.4.0-65-generic, microcode: 0x4003003, CPU governor: performance.

Software: scikit-learn 0.24.1, pandas 1.2.2, XGBoost 1.3.3, Python 3.9.7, scikit-learn-intelxx 2021.2, modin 0.8.3, omniscidbe v5.4.1.

Higher is better
E2E Recommendation System (E2E DIEN)

E2E DIEN: A representative E2E inference workload of recommendations that provides the capability of estimating use clicks at scale.
Optimized frameworks take advantage AVX512 and AMX instructions.

Conversion to Ops to Intel oneDNN (MKL) Ops optimized for Intel HW

Operator Fusions enable even more speedup & efficient use during runtime execution.

- GRU(MklFusedMatmul)
- Attention layer(MklFusedMatmul)
E2E Recommendation System (E2E DIEN)

E2E Recommendation system got significant improvement of ~23x over baseline by applying multiple optimization strategies together.
E2E NLP Document-level Sentiment Analysis (DLSA)

A representative E2E Fine-Tuning & Inference NLP workload with Sentiment Analysis Task on Huggingface Transformer API

Preprocessing

Tokenization + Feature Extraction

Model Loading

Initialize Transformer for fine-tuning

Deep Learning

Inference

Fine-Tuning

Load Training Dataset

Load Pretrained Model

Setup Sentiment Analysis Task Classifier

Load Fine-Tuned Model

Fine-tuning (Batched) Loop

Output Positive / Negative Sentiment & Accuracy

Results evaluation

Storage

IMDB or SST-2 Dataset

BERT Base or Large

Fine-Tuned DLSA Model

Output

Dataset

Model

Storage

Load Inference Dataset

Tokenization + Feature Extraction

Load Fine-Tuned Model

Inference (Batched) Loop

Perform Sentiment Analysis Task Classification

Post processing

Results

E2E NLP Document-level Sentiment Analysis (DLSA)

A representative E2E Fine-Tuning & Inference NLP workload with Sentiment Analysis Task on Huggingface Transformer API

Transformer

PYTORCH

TensorFlow

Transformer
Optimized frameworks take advantage AVX512 and AMX instructions.

Conversion to Ops to Intel oneDNN (MKL) Ops optimized for Intel HW

Operator Fusions enable even more speedup & efficient use during runtime execution.

- Gelu (Bunch of smaller ops fused to Gelu Op tf.nn.gelu)
- Fusion of MatMul + BiasAdd + Gelu(MklFusedMatmul)
- Fusion of BatchMatMul + Mul + Add(MklBatchMatmul)
- LayerNormalization (Bunch of smaller ops for keras layernorm api)(MklLayernorm)
Fine-Tuning Explained with Context to Hugging face DLSA

- Process of using a pretrained model, trained on a different source dataset, to train (modify training parameters) of a new target model (fine-tune it) with a different task (output layer) using a different target dataset.
- In DLSA we use BERT model for Masked-Language-Modeling task pretrained on large corpus of English data, to fine tune a new BERT model for sentiment analysis task on SST-2 or IMDB dataset.
E2E NLP Sentiment Analysis Inference (DLSA)

**Optimized AI/Analytics Packages**

- **Parallelize**
  - single instance to multi-instance
  - single node to multi-node

- **Vectorize**
  - High vector efficiency through oneDNN optimization library.

- **Graph Opt**
  - OP fusion
  - Batch normalization
  - Weight Caching

- **Precision**
  - Auto Mix Precision (BF16)
  - 8-bit Quantization using INC.

**Powered by Intel optimized AI/Analytics Packages**

E2E DLSA got significant total improvement of ~3.36x over baseline by applying multiple optimization strategies together.
Call To Action

- Efficient AI happens when AI frameworks and libraries, system tuning, model & hyperparameter optimization and run-time parameter tuning all work together cohesively
  - A clear path towards end-to-end AI performance roofline is achievable

- Every phase of an end-to-end AI pipeline needs to be efficient and optimized to realize an overall effective AI solution

- Performance acceleration with “optimization toolbox” strategies on CPUs brings significant boost in efficiency for end-to-end AI pipelines