End-to-End Optimization of AlphaFold2 on Intel Architecture

Luke Ge, Intel AI Technical Solution Specialist
Luke.ge@intel.com
AI Needed for Processing Huge Sequence Data

- X-Ray/ Cryo-EM: months ~ years per sequence
- 566,996 sequences in SwissProt wait for analysis
- Traditional methods (CADD) cannot handle huge data reliably
- More powerful protein folding tools are needed

DeepMind AlphaFold2 is a fast folding algorithm
- Two areas to be improved:
  - GPU Memory limits the sequence length (<1000aa)
  - Original code not optimized for CPU

AlphaFold2 Inference Performance Claim

- We optimized AlphaFold2 inference pipeline on Xeon ICX server (precision=FP32)
  - **4.01x** throughput on ICX8358 2S vs. single Nvidia A100 GPU card
  - we improve AlphaFold2 performance by **23.11x**: vs stock AlphaFold2
    - 4.56x pipeline throughput by multi-instance with PMEM
    - 5.05x pipeline throughput by model and op optimizations
AlphaFold2 Pipeline Overview

**Preprocessing**
- Get structure fragments templates from similar known protein sequences as references
- Search for similar structures by query in MSA reference database

**DL Model Inference**
- Transformers to predict the graph representations of 3D structures
- Template MSA template pair representations
- Evoformer (x48)
- Predicted LDDT
- Experimentally Resolved
- Distogram
- MaskedMSA
- Structure Module

**Postprocessing**
- Refine the 3D structure parameters
- Post-refine by Amber force field
- pdb fixer

**Compute time % on Xeon SP (stock AlphaFold2)**

- **Preprocessing**: 15.3%
- **DL Model Inference**: 84.1%
- **Postprocessing**: 0.6%

**Hardware**
- AVX-512
- oneAPI AI Analytics
- AVX-512
High Throughput Optimization of AlphaFold2 Pre-processing
Use AVX-512 to increase throughput (icc options)

-`-O3 -no-prec-div -march=icelake-server`

-`numactl –C $core_ids -m $socket_id $command`
-`mpirun –np $nranks –map-by ppr:$core_per_instance:socket:pe=$total_number_core_per_socket $command`

After

Before
Optimized Pre-processing

Before: 1 instances * 4 seqs
After: 64 instances * 1 seq

Test data: mm cif_3geh.fa, seq length = 765
Compute nodes: Intel® Xeon 8358x2, 512GB DDR4+4TB PMEM, 25GbE NIC
Ubuntu 20.04 server, Python 3.9.7, hmmer 3.3.2
Multi-instance Challenge: Memory

AlphaFold2 is memory hungry

Memory footprint in Pipeline (GB, per instance, amino acids length=765)

Peak memory usage of a pre-processing instance on CPU

49.7
9.1
36.9
37.2
37.2
4.2

Memory Footprint vs. Throughput
With multi-instance (amino acids length=765)

4.56x boost from multi-instances with large memory

Each instance needs ~37GB for DL inference on CPU/GPU

Throughput (samples per day)

0.0
20.0
40.0
60.0
80.0
100.0
120.0

0
500
1000
1500
2000
2500
3000
3500

0
49.7GB
198.8GB
397.6GB
795.2GB
1590.4GB
3180.8GB

Multi-instance Challenge: Memory

4x Preprocessing, 4x DL inference, 4x Postprocessing
8x Preprocessing, 8x16 seq batches DL inference, 8x Postprocessing
16x Preprocessing, 4x4 seq batches DL inference, 16x Postprocessing
32x Preprocessing, 4x8 seq batches DL inference, 32x Postprocessing
64x Preprocessing, 4x16 seq batches DL inference, 64x Postprocessing

AlphaFold2 is memory hungry
**BREAKTHROUGH MEMORY INNOVATION**
- Affordable alternative to DRAM
- Improve TCO
- On-module encryption

**INFRASTRUCTURE CONSOLIDATION**
- Increase memory size (128/256/512GB)
- Consolidate workloads
- Scale up to scale out

**PERFORMANCE AND PERSISTENCE**
- Break IO bottlenecks
- Faster recovery
- High speed storage

INTEL.COM/OPTANE DC PERSISTENT MEMORY
## AlphaFold2 TCO: Xeon vs. Nvidia Tesla A100

### Normalized Performance per Dollar (Amino acids length=765)

<table>
<thead>
<tr>
<th></th>
<th>Xeon SP 6330 2S + 8x Tesla A100-40GB</th>
<th>Xeon SP 8358 2S with PMEM (16x64GB DDR4+16x256GB BPS)</th>
<th>Xeon SP 8358 2S with DDR4 (26*128GB DDR4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Inspur NF5468M6-P</td>
<td>Inspur NF5280M6</td>
<td>Inspur NF5280M6</td>
</tr>
<tr>
<td>Processor</td>
<td>3rd Gen Intel® Xeon® Scalable</td>
<td>3rd Gen Intel® Xeon® Scalable</td>
<td>3rd Gen Intel® Xeon® Scalable</td>
</tr>
<tr>
<td></td>
<td>processors 6330 x2</td>
<td>processors 8358 x2</td>
<td>processors 8358 x2</td>
</tr>
<tr>
<td>GPU</td>
<td>Nvidia A100 PCIe Gen4 40 GB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Memory</td>
<td>32GB DDR4 3200MHz RDIMM x16</td>
<td>32GB DDR4 3200MHz RDIMM x16 and 256GB Optane Intel</td>
<td>128GB DDR4 3200MHz RDIMM x32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMB1XXD256GPSU4 DCPMM Barlow Pass x16</td>
<td></td>
</tr>
<tr>
<td>I/O</td>
<td>Raid Cntrlr - Trinity Dunes RAID</td>
<td>Raid Cntrlr - Trinity Dunes RAID Adapter</td>
<td>Raid Cntrlr - Trinity Dunes RAID Adapter</td>
</tr>
<tr>
<td></td>
<td>Adapter</td>
<td>Intel RSP3TD160F x1</td>
<td>Intel RSP3TD160F x1</td>
</tr>
<tr>
<td>Storage</td>
<td>Solidigm Youngsville Refresh SSDSC2KB038T801 S4510 Series x1</td>
<td>Solidigm Youngsville Refresh SSDSC2KB038T801 S4510 Series x1</td>
<td>Solidigm Youngsville Refresh SSDSC2KB038T801 S4510 Series x1</td>
</tr>
<tr>
<td>Network</td>
<td>SND I350-AM2 RJ45 Dual Port PCI-E4X_1KM x1</td>
<td>SND I350-AM2 RJ45 Dual Port PCI-E4X_1KM x1</td>
<td>SND I350-AM2 RJ45 Dual Port PCI-E4X_1KM x1</td>
</tr>
<tr>
<td>PSU</td>
<td>2+2 redundant, 4x 3000W 80Plus Platinum hot-swap PSU</td>
<td>(1+1) 1200W Platinum</td>
<td>(1+1) 1200W Platinum</td>
</tr>
<tr>
<td>Reference Price (USD)</td>
<td>113,150</td>
<td>19,832</td>
<td>48,650</td>
</tr>
</tbody>
</table>

### TCO Analysis Configuration

<table>
<thead>
<tr>
<th></th>
<th>Nvidia Tesla A100 with Xeon SP</th>
<th>Xeon SP with DDR4+PMEM</th>
<th>Xeon SP with DDR4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Inspur NF5468M6-P</td>
<td>Inspur NF5280M6</td>
<td>Inspur NF5280M6</td>
</tr>
<tr>
<td>Processor</td>
<td>3rd Gen Intel® Xeon® Scalable</td>
<td>3rd Gen Intel® Xeon® Scalable</td>
<td>3rd Gen Intel® Xeon® Scalable</td>
</tr>
<tr>
<td></td>
<td>processors 6330 x2</td>
<td>processors 8358 x2</td>
<td>processors 8358 x2</td>
</tr>
<tr>
<td>GPU</td>
<td>Nvidia A100 PCIe Gen4 40 GB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Memory</td>
<td>32GB DDR4 3200MHz RDIMM x16</td>
<td>32GB DDR4 3200MHz RDIMM x16 and 256GB Optane Intel NMB1XXD256GPSU4 DCPMM Barlow Pass x16</td>
<td>128GB DDR4 3200MHz RDIMM x32</td>
</tr>
<tr>
<td>I/O</td>
<td>Raid Cntrlr - Trinity Dunes RAID Adapter</td>
<td>Raid Cntrlr - Trinity Dunes RAID Adapter</td>
<td>Raid Cntrlr - Trinity Dunes RAID Adapter</td>
</tr>
<tr>
<td></td>
<td>Intel RSP3TD160F x1</td>
<td>Intel RSP3TD160F x1</td>
<td>Intel RSP3TD160F x1</td>
</tr>
<tr>
<td>Storage</td>
<td>Solidigm Youngsville Refresh SSDSC2KB038T801 S4510 Series x1</td>
<td>Solidigm Youngsville Refresh SSDSC2KB038T801 S4510 Series x1</td>
<td>Solidigm Youngsville Refresh SSDSC2KB038T801 S4510 Series x1</td>
</tr>
<tr>
<td>Network</td>
<td>SND I350-AM2 RJ45 Dual Port PCI-E4X_1KM x1</td>
<td>SND I350-AM2 RJ45 Dual Port PCI-E4X_1KM x1</td>
<td>SND I350-AM2 RJ45 Dual Port PCI-E4X_1KM x1</td>
</tr>
<tr>
<td>PSU</td>
<td>2+2 redundant, 4x 3000W 80Plus Platinum hot-swap PSU</td>
<td>(1+1) 1200W Platinum</td>
<td>(1+1) 1200W Platinum</td>
</tr>
<tr>
<td>Reference Price (USD)</td>
<td>113,150</td>
<td>19,832</td>
<td>48,650</td>
</tr>
</tbody>
</table>
AlphaFold2 DL Model
Inference Optimization
Migrate AlphaFold2 to PyTorch

- Stock AlphaFold2 models are based on JAX
  - leverage google’s XLA compile
  - not optimized on CPU yet
- Migrate to PyTorch
  - models (embedding, Evoformer, heads) manually rewritten by PyTorch
  - model correctness validated: <1.63% error in output
  - leverage PyTorch’s JIT optimization

Model inference latency Before/After migration (seconds, single instance, amino acids length=765)

- Intel internal private repo at this moment
- Plan to be open-source in Intel Model Zoo
Memory Bottleneck in Attention Module (ExtraMsaStack)

```
self.attention(q_data, m_data, bias, nonbatched_bias)
```

**Original**
- 5120x764x64 (930MB)
- 5120x1x1x64 (1.25MB)
- 5120x1x764x64 (930MB)
- 320x1x1x64 (80KB)
- 320x1x764x64 (59.69MB)

**Optimized**
- 1.74x boost in attention unit
- 21.51% boost in Embedding model

```python
def slice_attention(self, q_data, m_data, bias, nonbatched_bias):
    # avoiding huge memory cost
    # threshold is adjustable
    threshold = 1000
    unit = 320 # unit is adjustable
    if q_data.size()[0] > threshold:
        res = torch.ones_like(q_data)
        for i in range(q_data.size()[0] // unit):
            q_sub_data = q_data[unit*i:unit*(i+1)]
            m_sub_data = m_data[unit*i:unit*(i+1)]
            bias_sub = bias[unit]
            res[unit*i:unit*(i+1)] = self.attention(q_sub_data, m_sub_data, bias_sub, nonbatched_bias)
        return res
    else:
        return self.attention(q_data, m_data, bias, nonbatched_bias)
```
Operation Fusion in Evoformer Module

An Evoformer Block

- Fuse einsum and add by using oneAPI
- Fusion is available with oneDNN 2.6

Original

Optimized

<table>
<thead>
<tr>
<th>Name</th>
<th>Self CPU %</th>
<th>CPU total %</th>
</tr>
</thead>
<tbody>
<tr>
<td>aten::einsum</td>
<td>0.07%</td>
<td>42.23%</td>
</tr>
<tr>
<td>aten::bmm</td>
<td>26.73%</td>
<td>37.51%</td>
</tr>
<tr>
<td>quantized::linear_dynamic</td>
<td>20.15%</td>
<td>20.16%</td>
</tr>
<tr>
<td>aten::clone</td>
<td>0.08%</td>
<td>16.45%</td>
</tr>
<tr>
<td>aten::copy_</td>
<td>16.35%</td>
<td>16.35%</td>
</tr>
<tr>
<td>aten::add</td>
<td>12.62%</td>
<td>12.62%</td>
</tr>
<tr>
<td>aten::softmax</td>
<td>0.00%</td>
<td>8.76%</td>
</tr>
<tr>
<td>aten::_softmax</td>
<td>8.76%</td>
<td>8.76%</td>
</tr>
</tbody>
</table>

Large Tenors
(~140MB per tensor)

Batch Matrix Multiplication (BMM)  Add

6.03x boost in Einsum+Add Ops unit test
10.02% boost in Evoformer model
5.0% boost in AlphaFold2 pipeline
### AlphaFold2 Pipeline Inference Latency (Single Instance)

#### Before Optimization

**Total:** 18964.4 seconds

- Preprocessing: 15.37%
- DL Inference: 84.02%
- Postprocessing: 0.61%

#### After Optimization

**Total:** 3759.0 seconds

- Preprocessing: 71.72%
- DL Inference: 25.21%
- Postprocessing: 3.07%

5.05x boost

Sample length = 765 amino acids
Pre-processing = MSA + TemplateMatch
DL inference = Embedding + Evoformer + Heads
Post-processing = Amber
Accuracy Comparison: Optimized vs. Original

- Input: 206aa
  - 2S-ICX8358

- Input: 765aa
  - 2S-ICX8358

- MatchMaker
- Gold: original
- Blue: optimized
Summary

- Intel optimized AlphaFold2 inference on Xeon platform
  - Memory: larger memory space (TBs), Optane Persistent Memory
  - Compute: AVX-512 (advanced vector extension), AMX (advanced matrix extension)
  - Communication: easy scaling on cluster
  - oneAPI: free AI software stack
- CPU is a better platform for AlphaFold2 inference
  - Intel Xeon CPU is 4.01x faster than Nvidia A100, 2.87x better Perf/$
  - CPU processes longer sequences than GPU
- The optimized code will be publicly available
### Key hardware configuration

<table>
<thead>
<tr>
<th>Component</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>2x 3rd Gen Intel® Xeon® Scalable processors 8358 or higher bin</td>
</tr>
<tr>
<td>DDR4 Memory</td>
<td>16x 32GB DDR4 3200MHz RDIMM</td>
</tr>
<tr>
<td>PMEM Memory</td>
<td>16x 256GB PMEM (BPS)</td>
</tr>
<tr>
<td>Storage</td>
<td>Intel® Optane SSD 3.8TB+</td>
</tr>
</tbody>
</table>

### Key Software configuration

<table>
<thead>
<tr>
<th>Component</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation System</td>
<td>CentOS 7.x/Ubuntu 20.04 or later version</td>
</tr>
<tr>
<td>AI Framework</td>
<td>PyTorch 1.11.0 Intel PyTorch Extension 1.11.0</td>
</tr>
<tr>
<td>Library and compiler</td>
<td>Intel® oneAPI Base Toolkit 2022.1.2 or later version</td>
</tr>
<tr>
<td>Python</td>
<td>Intel® distribution for Python integrated in Intel® oneAPI Base Toolkit</td>
</tr>
</tbody>
</table>

Luke.ge@intel.com