Microsoft Research

Antares for SYCL

A Tool for Cross-Platform Kernel Optimization

Presenter - *Wei Cui*

MSRA System Research Asia (Beijing)

https://github.com/microsoft/antares

Antares: Code Generation + Multi-Backends + Tuning Stack

Parallel CPU Backend

Clang

SYCL Backend

oneAPI

SYCL

codeplay

ComputeCpp

(experimental GPU support?)

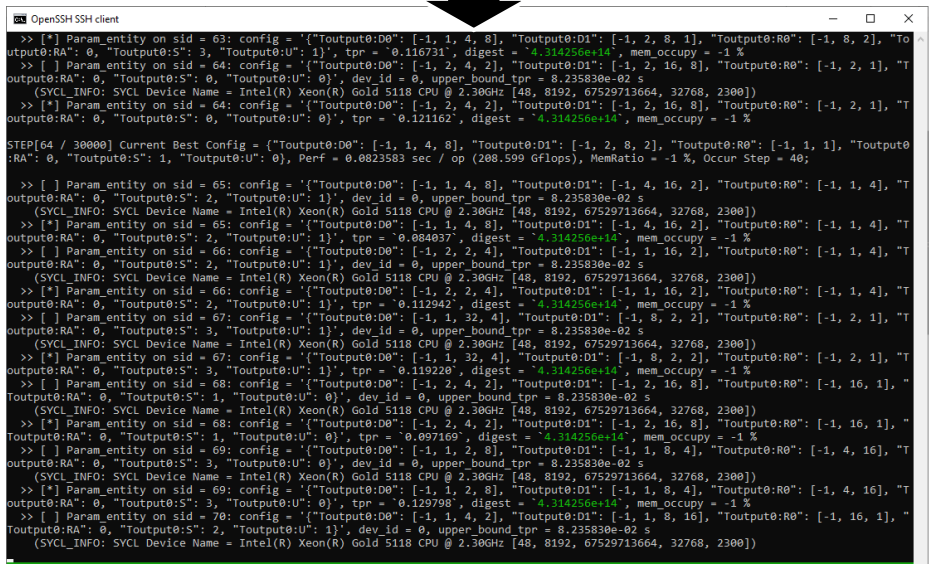
Antares

Input and Output for Antares

1. Describe What to Compute:
(as input) →

*output0[N, M] +=! input0[N, K] * B[K, M]*
(GEMM operator based on Antares IR)

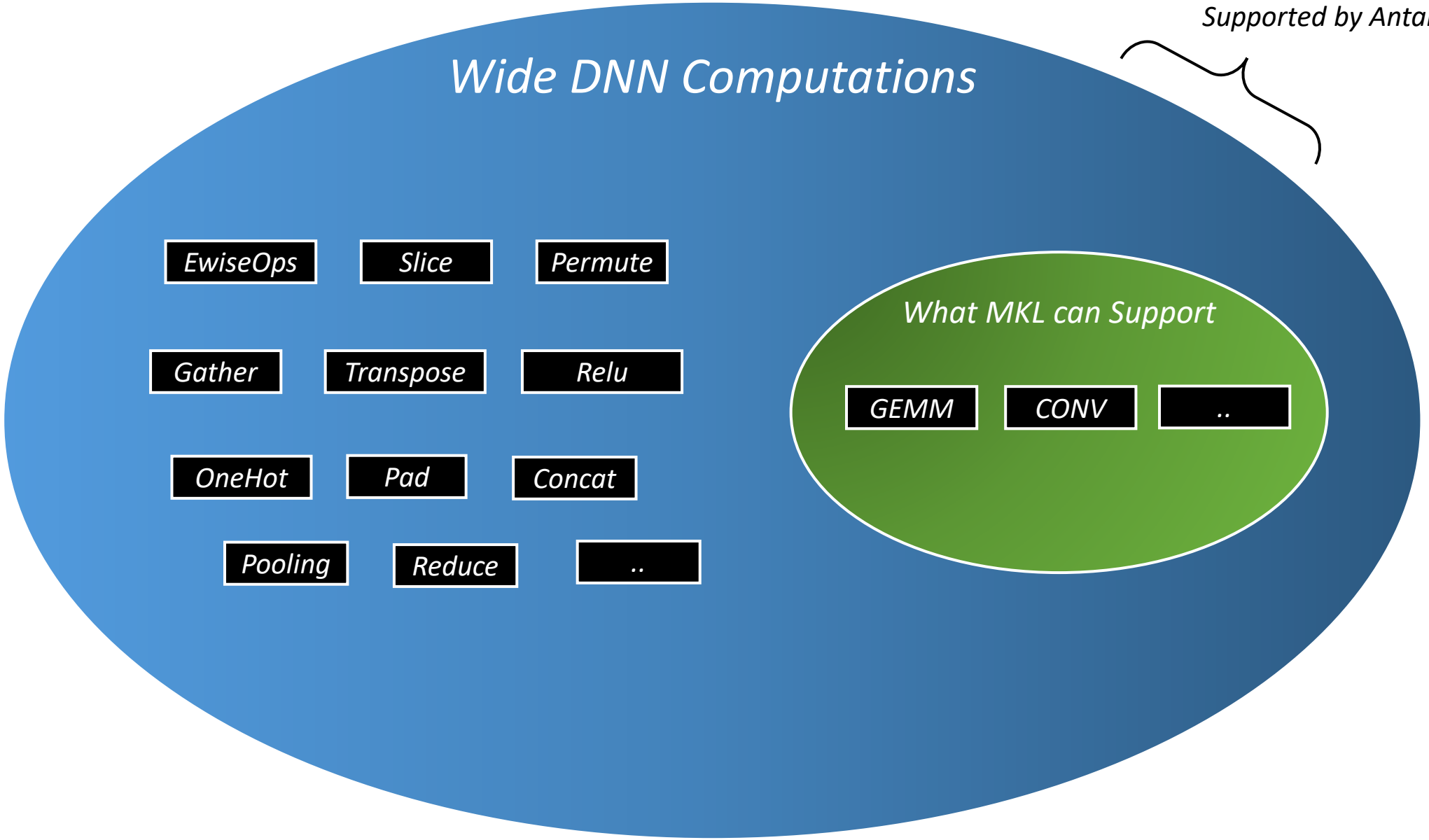
2. Auto-Tune Progress:
(optimization stage) →



3. How does "Output Code" looks like:
(as output) ↓

```
const auto* input0 = ..;
const auto* input1 = ..;
auto* output0 = ..;

using namespace cl::sycl;

q->submit([&](auto &cgh) {


  cgh.parallel_for(cl::sycl::nd_range<3>(cl::sycl::range<3>(1, 512, 1024), cl::sycl::range<3>(1, 1, 1)), [=](cl::sycl::nd_item<3> _item) {

    const int blockIdx_x = _item.get_group(2), blockIdx_y = _item.get_group(1), blockIdx_z = _item.get_group(0), threadIdx_x = _item.get_local_id(2), threadIdx_y = _item.get_local_id(1), threadIdx_z = _item.get_local_id

    ...
    output0[((((((int)blockIdx_x) * 512) + ((int)blockIdx_y)))] = (input0[((((((int)blockIdx_x) * 512) + ((int)blockIdx_y)))] + input1[((((((int)blockIdx_x) * 512) + ((int)blockIdx_y)))]);
    ...
  });
});
```

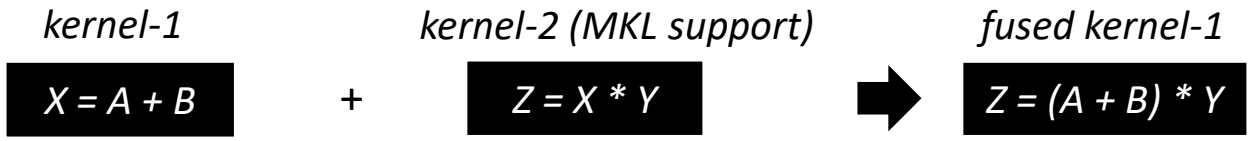
Computation Scopes that Antares can Optimize



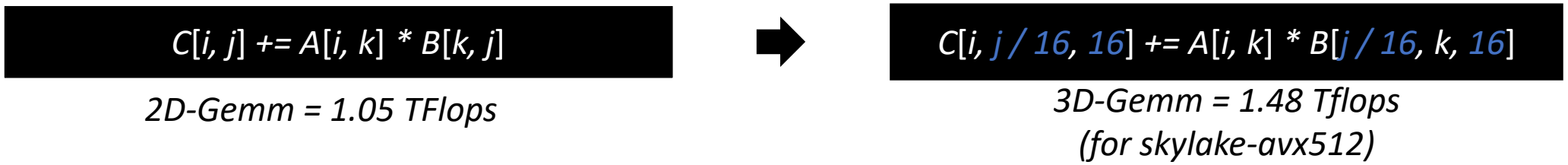
Is it Necessary to Optimize MKL Operators using Antares?

e. g. Following Cases that MKL isn't helpful ——

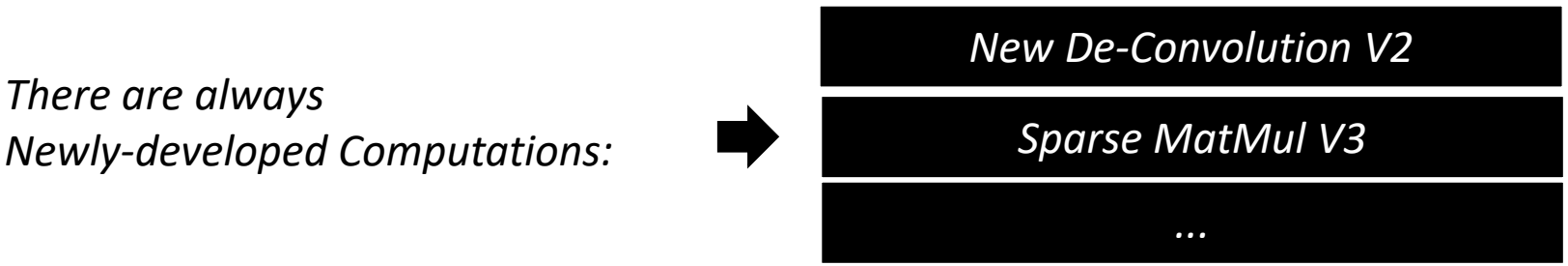
1. Inline Fusion / Rammer Fusion:



2. Standard Layout → Preferred Layout:



3. Other Non-Standard Computation Requirement:



Antares Example for Tensorflow (Optimizing MNIST)

```
import tensorflow as tf
from tensorflow.contrib import antares

def create_param(name, shape):
  return tf.get_variable(name, shape, tf.float32, initializer=tf.initializers.ones(tf.float32))

input0 = create_param('input0', [64, 28 * 28])
w0, b0 = create_param('w0', [28 * 28, 512]), create_param('b0', [512])
w1, b1 = create_param('w1', [512, 512]), create_param('b1', [512])
w2, b2 = create_param('w2', [512, 10]), create_param('b2', [10])

output0 = antares.make_op(ir='''
  data_0[N, M] +=!  data[N, K] * w_0[K, M];
  data_1[N, K] =   (data_0[N, K] + bias_0[K]).call(`max`, [0.0]);  -- fused
  data_2[N, M] +=!  data_1[N, K] * w_1[K, M];
  data_3[N, K] =   (data_2[N, K] + bias_1[K]).call(`max`, [0.0]);  -- fused
  data_4[N, M] +=!  data_3[N, K] * w_2[K, M];
  data_5[N, K] =   (data_4[N, K] + bias_2[K]);                     -- fused
''', feed_dict={
  'data': input0, 'w_0': w0, 'w_1': w1, 'w_2': w2, 'bias_0': b0, 'bias_1': b1, 'bias_2': b2,
}).tune(step=200, use_cache=True, timeout=600).emit()

config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
  sess.run(tf.global_variables_initializer())
  print(sess.run(output0))
```

Example Works for:

1) Extend Clang Op for Intel-TF;
2) Extend SYCL Op for Intel-TF;
3) Extend CUDA Op for TF-CUDA;
4) Extend ROCM Op for TF-AMDGPU;

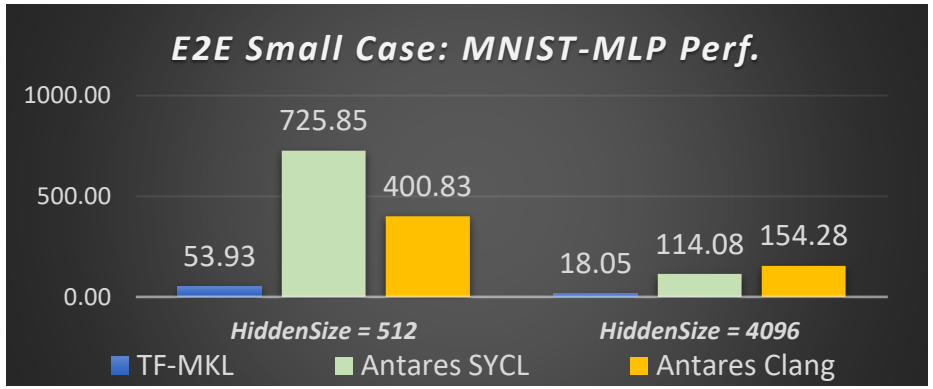
Result: Intel-MKL v.s. ONNX-MLAS v.s. Antares SYCL v.s. Antares CLANG

Microsoft Research

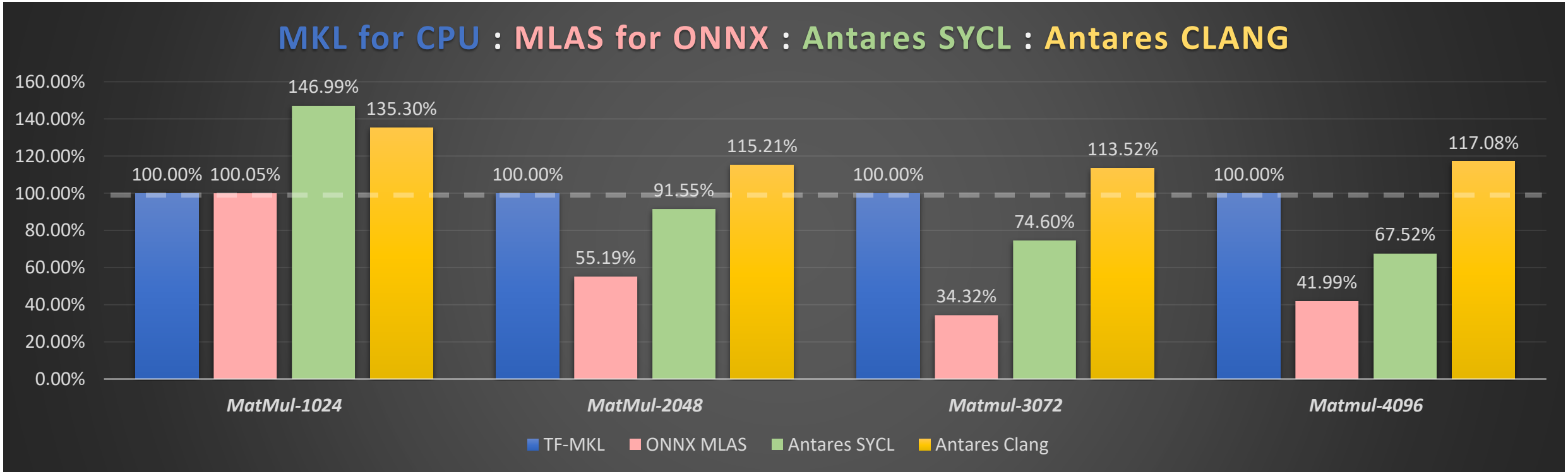
CPU Type: *Intel(R) Xeon(R) Gold 5118 CPU (12 core x 2 socket)*
DPC++ Version: *Intel OneAPI - 2021.2*

(All have AVX512 enabled)

**** Small Workloads:*** *Antares SYCL > Antares Clang > MKL ≈ ONNX MLAS*
**** Large Workloads:*** *Antares Clang ≈ MKL > ONNX MLAS > Antares SYCL*



Antares SYCL: ✔ *Small Case* ✔ *Fusion* ✔ *No-Eigen*



https://github.com/microsoft/antares