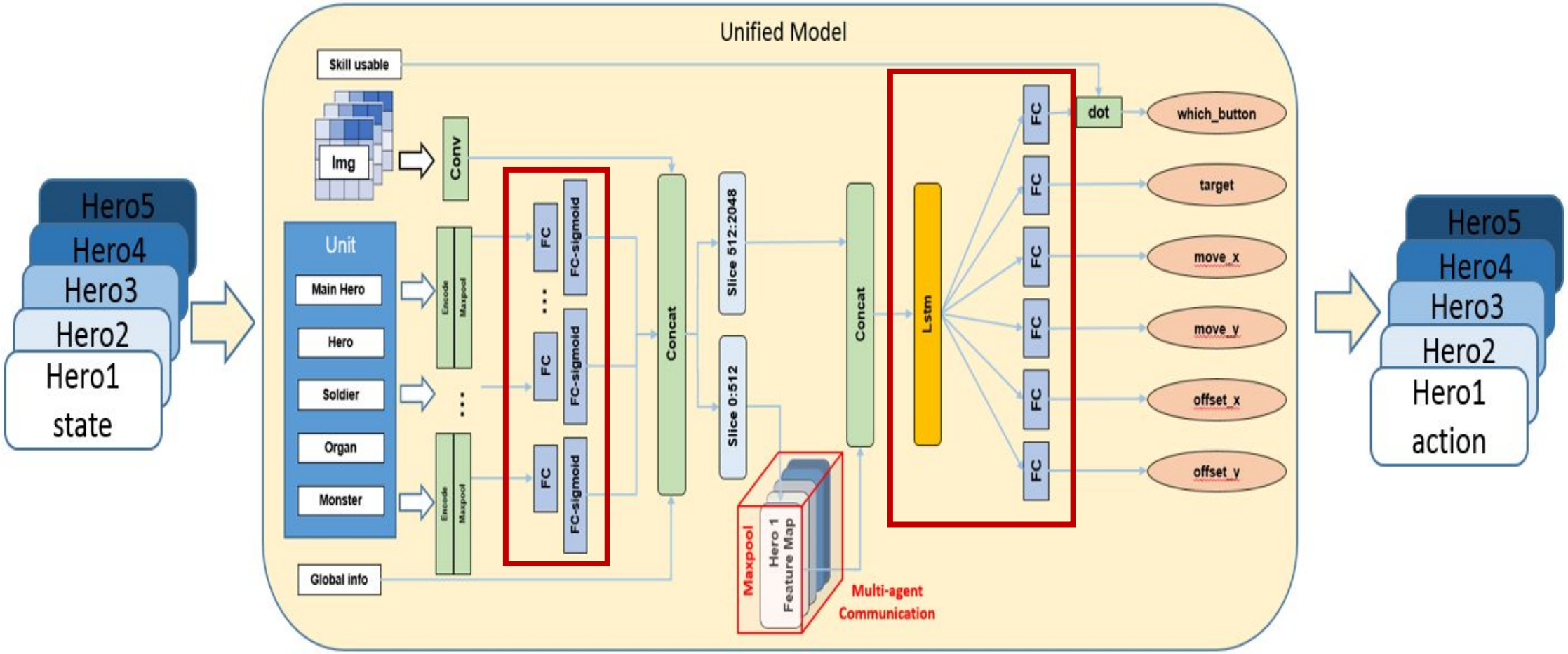
Tencent AI Lab

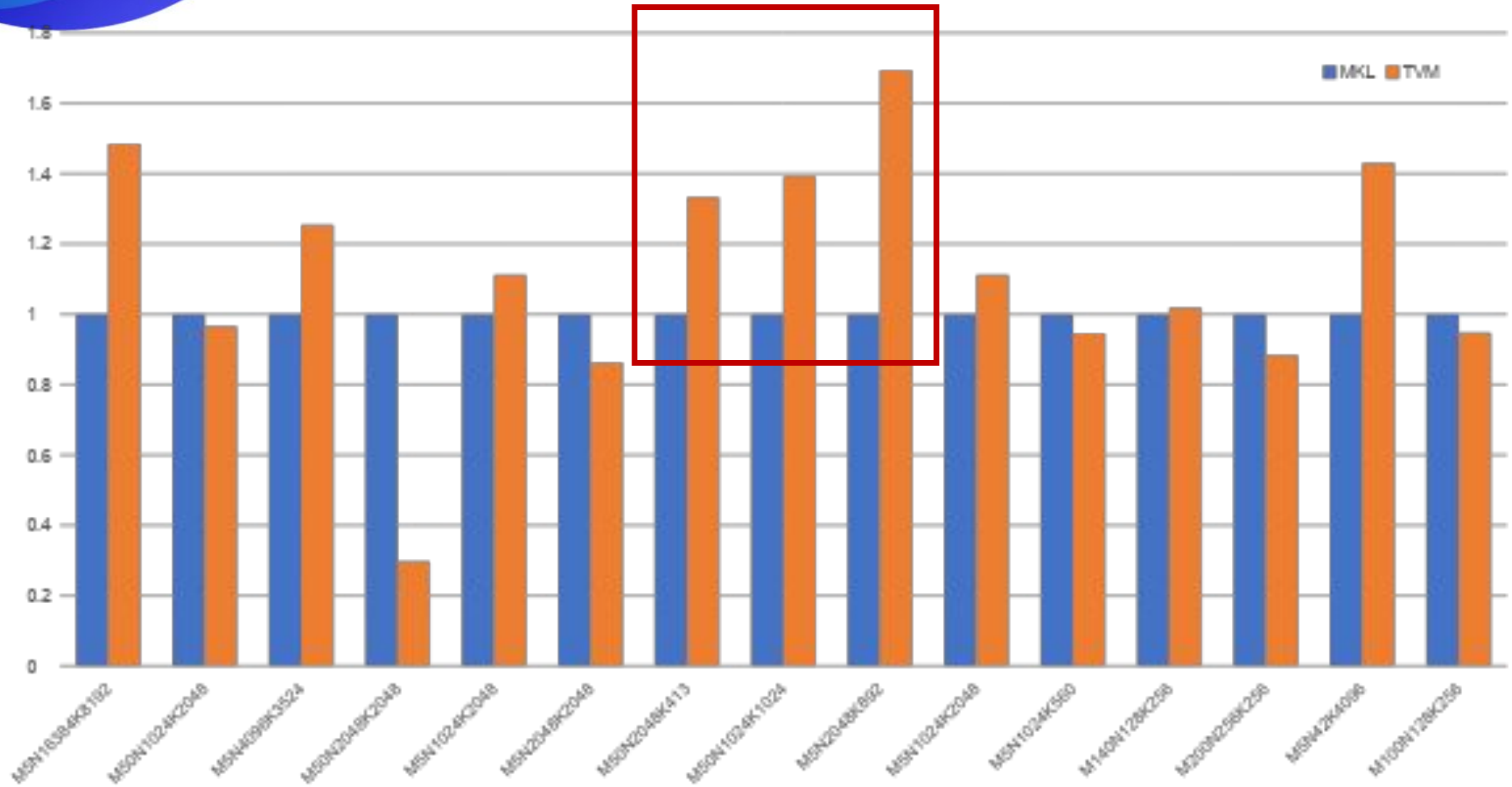
Extremely Fast GEMM on AVX512 CPUs Combining TVM and XSMM

Honglin Zhu
Tencent AI Lab

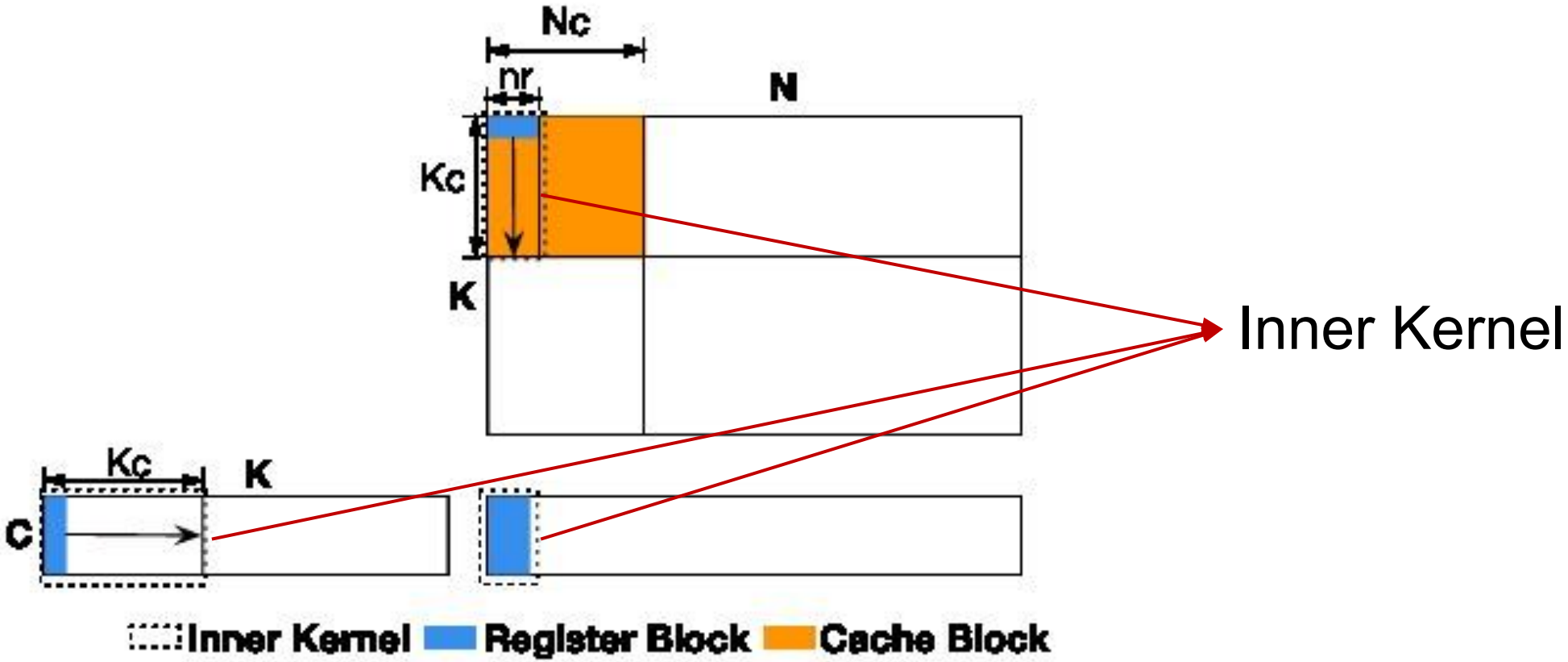
Wukong AI Model



MKL vs TVM


MKL
TVM
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
M5N16384K8192
M50N1024K2048
M5N4096K3584
M50N2048K2048
M5N1024K2048
M5N2048K2048
M50N2048K413
M50N1024K1024
M5N2048K892
M5N1024K2048
M5N1024K560
M140N128K256
M3008256K256
M5N42K4096
M1050N128K256

Performance Bottleneck



Solution: Tensorize + Micro-Kernel

Tencent AI Lab

```
  for (y: int32, 0, 34) {
    packedB[ramp((y*64), 1, 64)] = (float32x64*)B_2[ramp((y*64), 1, 64)]
  }
  for (x.outer.y.outer.fused: int32, 0, 928) {
    C.global[ramp(0, 1, 64)] = broadcast(0f32, 64)
    C.global[ramp(64, 1, 64)] = broadcast(0f32, 64)
    C.global[ramp(128, 1, 64)] = broadcast(0f32, 64)
    C.global[ramp(192, 1, 64)] = broadcast(0f32, 64)
    for (k.inner: int32, 0, 34) {
      C.global[ramp(0, 1, 64)] = ((float32x64*)C.global[ramp(0, 1, 64)] + (broadcast((float32*)A_2[((x.outer.y.outer.fused*136) + k.inner)], 64)*(floa
2x64*)packedB[ramp((k.inner*64), 1, 64)]))
      C.global[ramp(64, 1, 64)] = ((float32x64*)C.global[ramp(64, 1, 64)] + (broadcast((float32*)A_2[(((x.outer.y.outer.fused*136) + k.inner) + 34)],
)*(float32x64*)packedB[ramp((k.inner*64), 1, 64)]))
      C.global[ramp(128, 1, 64)] = ((float32x64*)C.global[ramp(128, 1, 64)] + (broadcast((float32*)A_2[(((x.outer.y.outer.fused*136) + k.inner) + 68)]
64)*(float32x64*)packedB[ramp((k.inner*64), 1, 64)]))
      C.global[ramp(192, 1, 64)] = ((float32x64*)C.global[ramp(192, 1, 64)] + (broadcast((float32*)A_2[(((x.outer.y.outer.fused*136) + k.inner) + 102)
 64)*(float32x64*)packedB[ramp((k.inner*64), 1, 64)]))
    }
    C_2[ramp((x.outer.y.outer.fused*256), 1, 64)] = (float32x64*)C.global[ramp(0, 1, 64)]
    C_2[ramp(((x.outer.y.outer.fused*256) + 64), 1, 64)] = (float32x64*)C.global[ramp(64, 1, 64)]
    C_2[ramp(((x.outer.y.outer.fused*256) + 128), 1, 64)] = (float32x64*)C.global[ramp(128, 1, 64)]
    C_2[ramp(((x.outer.y.outer.fused*256) + 192), 1, 64)] = (float32x64*)C.global[ramp(192, 1, 64)]
  }
}
```

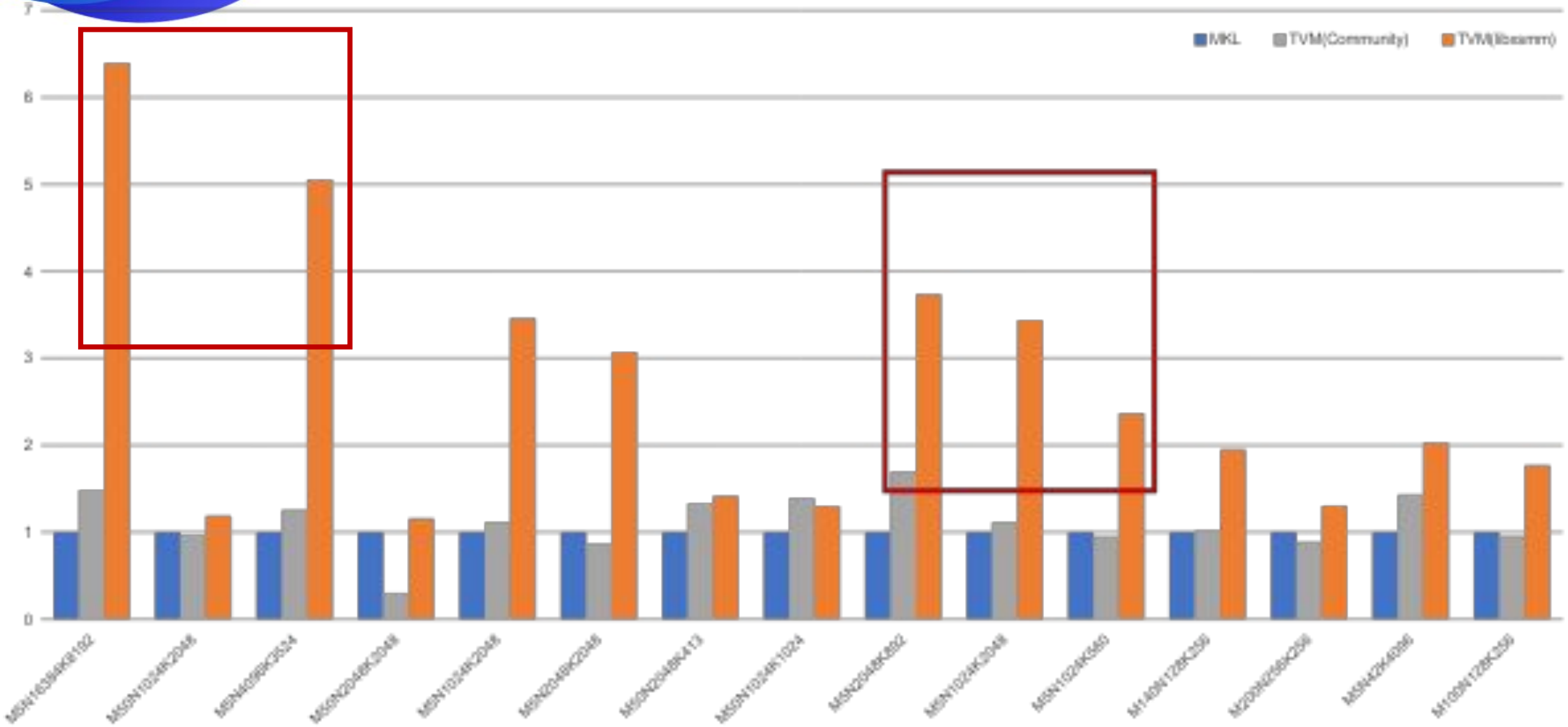
Tensorize

```
  for (k.outer: int32, 0, 128) {
    for (y.inner.outer: int32, 0, 16) "parallel" {
      for (k.inner.outer: int32, 0, 4) {
        if (@tir.likely((0 < k.outer), dtype=bool) || @tir.likely((0 < k.inner.outer), dtype=bool)) {
          @tir.tvm_call_packed("tvm.contrib.libxsmm.matmul", @tir.tvm_stack_make_array(A_2, @tir.tvm_stack_make_shape(5, 16, dtype=handle), @tir.tvm_sta
ck_make_shape(8192, 1, dtype=handle), 2, 0f32, ((k.outer*64) + (k.inner.outer*16)), dtype=handle), @tir.tvm_stack_make_array(B_2, @tir.tvm_stack_make_sh
ape(16, 1024, dtype=handle), @tir.tvm_stack_make_shape(16384, 1, dtype=handle), 2, 0f32, (((k.outer*1048576) + (k.inner.outer*262144)) + (y.inner.outer*
1024)), dtype=handle), @tir.tvm_stack_make_array(C_2, @tir.tvm_stack_make_shape(5, 1024, dtype=handle), @tir.tvm_stack_make_shape(16384, 1, dtype=handle
), 2, 0f32, (y.inner.outer*1024), dtype=handle), False, False, 1, 1, dtype=int32)
        } else {
          @tir.tvm_call_packed("tvm.contrib.libxsmm.matmul", @tir.tvm_stack_make_array(A_2, @tir.tvm_stack_make_shape(5, 16, dtype=handle), @tir.tvm_sta
ck_make_shape(8192, 1, dtype=handle), 2, 0f32, ((k.outer*64) + (k.inner.outer*16)), dtype=handle), @tir.tvm_stack_make_array(B_2, @tir.tvm_stack_make_sh
ape(16, 1024, dtype=handle), @tir.tvm_stack_make_shape(16384, 1, dtype=handle), 2, 0f32, (((k.outer*1048576) + (k.inner.outer*262144)) + (y.inner.outer*
1024)), dtype=handle), @tir.tvm_stack_make_array(C_2, @tir.tvm_stack_make_shape(5, 1024, dtype=handle), @tir.tvm_stack_make_shape(16384, 1, dtype=handle
), 2, 0f32, (y.inner.outer*1024), dtype=handle), False, False, dtype=int32)
        }
      }
    }
  }
```

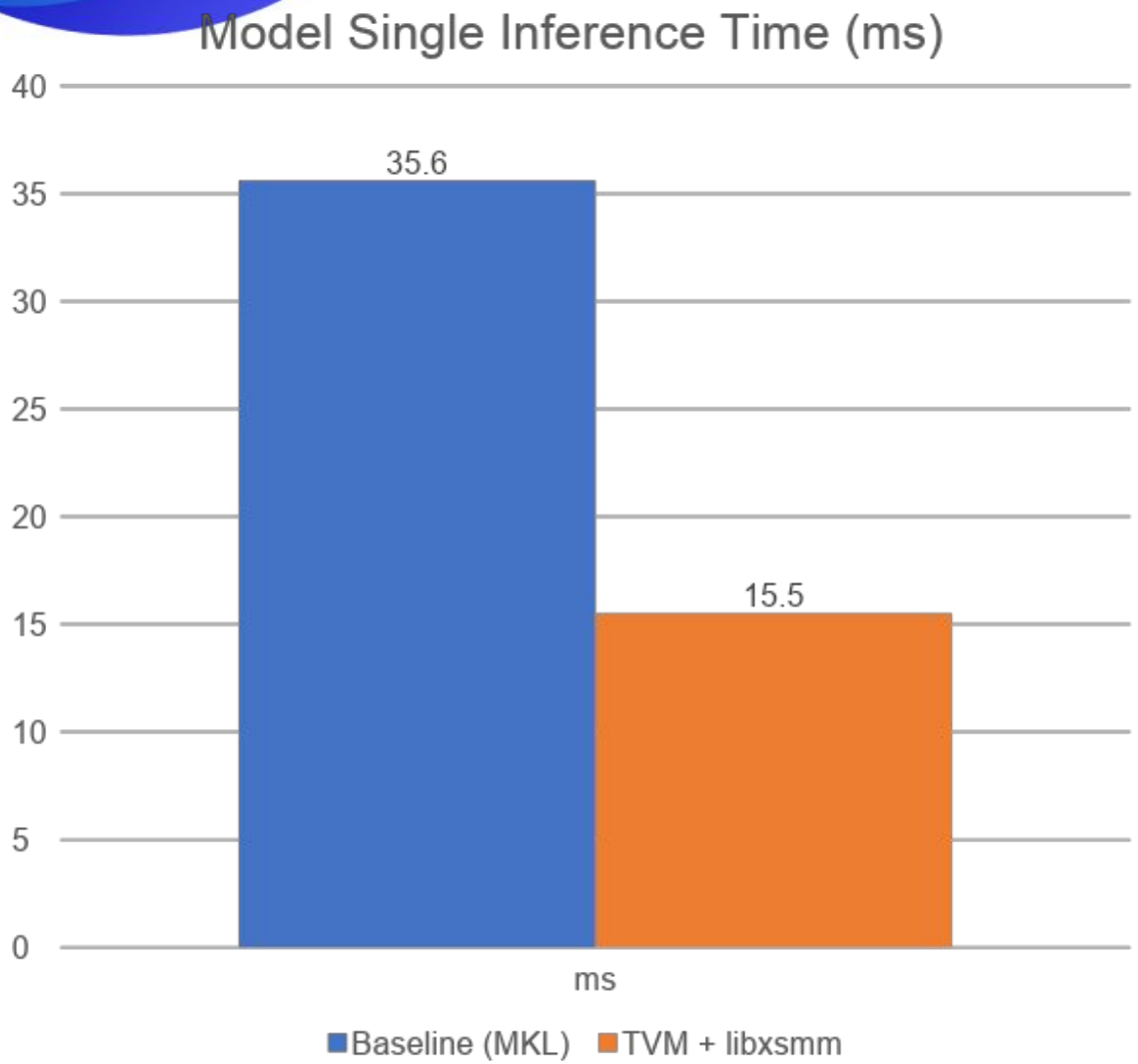
Libxsmm Micro Kernel

Result

Tencent AI Lab



Result



- Performance improved more than 2.3x.
- Server cost cut in half
- More than 1 million dollars saved every year

Tencent AI Lab

Thank You