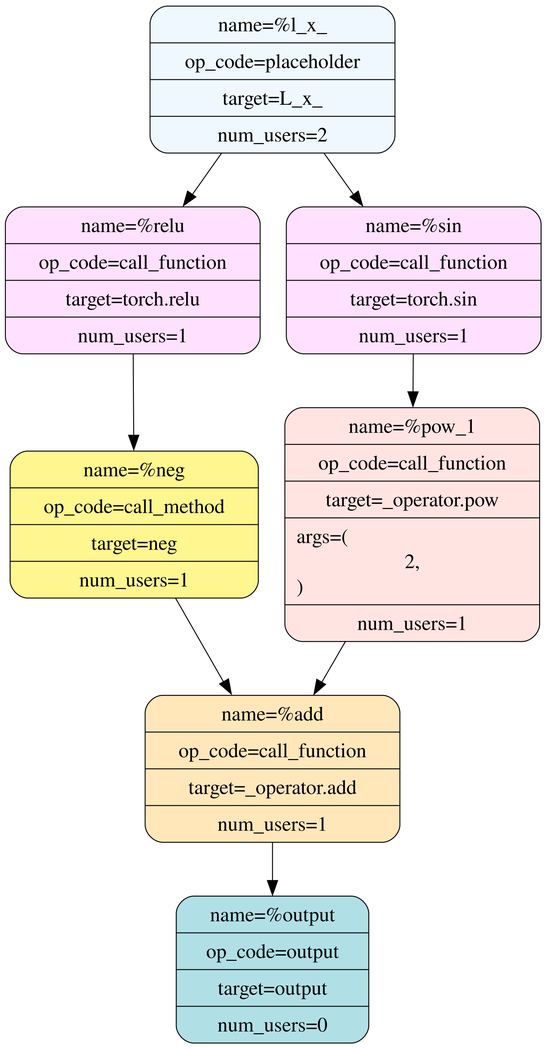
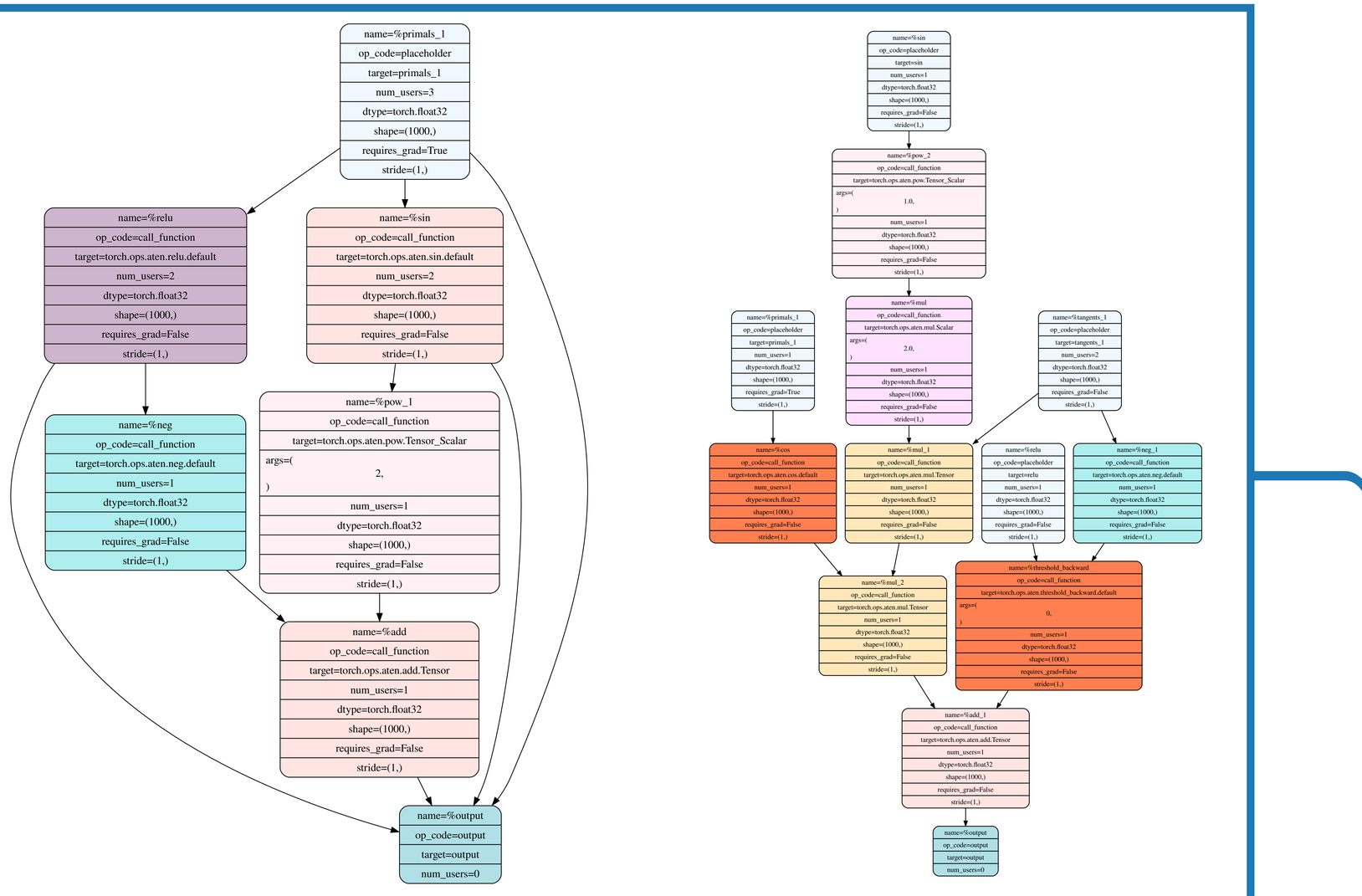
Graph Capture



Forward Graph

ATen AOT Automatic Differentiation



Forward Graph

Backward Graph

Code Generation

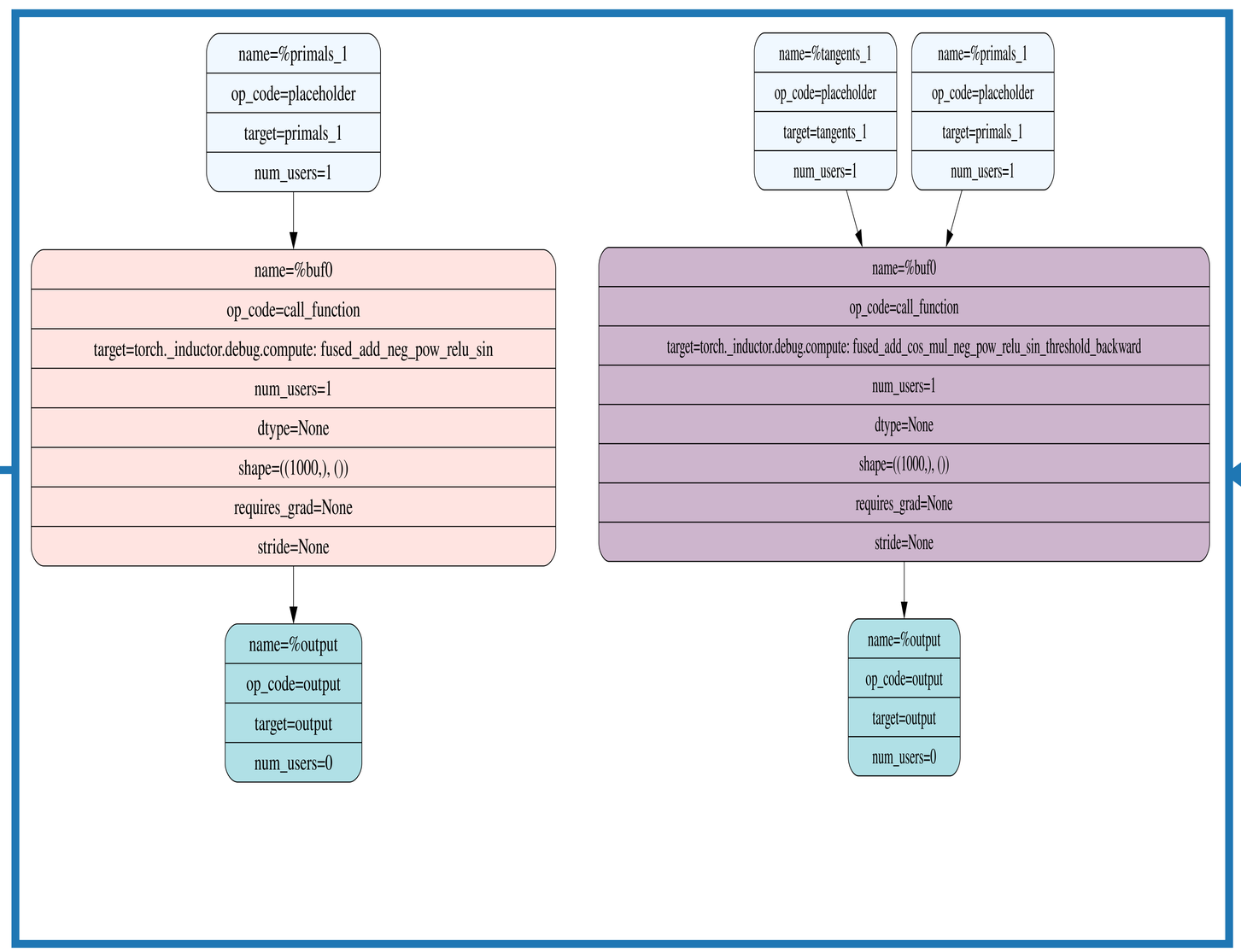
```
from torch import empty_strided, device
from torch._inductor.codecache import AsyncCompile
from torch._inductor.select_algorithm import extern_kernels

aten = torch.ops.aten
assert_size_stride = torch._C._dynamo.guards.assert_size_stride
reinterpret_tensor = torch.ops.inductor._reinterpret_tensor
async_compile = AsyncCompile()


cpp_fused_add_cos_mul_neg_pow_relu_sin_threshold_backward_0 = async_compile.cpp('''
#include "/tmp/torchinductor_pollakg/ib/cibrnuq56cxamjj4krp4zpjvsirbmlolpbnmomodzyd46huzhdw7.h"
extern "C" void kernel(const float* in_ptr0,
                       const float* in_ptr1,
                       float* out_ptr0)
{
    {
        for(long i0=static_cast<long>(0L); i0<static_cast<long>(1000L); i0+=static_cast<long>(8L))
        {
            auto tmp0 = at::vec::Vectorized<float>::loadu(in_ptr0 + static_cast<long>(i0));
            auto tmp1 = at::vec::Vectorized<float>::loadu(in_ptr1 + static_cast<long>(i0));
            auto tmp2 = tmp1.sin();
            auto tmp3 = at::vec::Vectorized<float>(static_cast<float>(2.0));
            auto tmp4 = tmp2 * tmp3;
            auto tmp5 = tmp0 * tmp4;
            auto tmp6 = tmp1.cos();
            auto tmp7 = tmp5 * tmp6;
            auto tmp8 = at::vec::clamp_min(tmp1, decltype(tmp1)(0));
            auto tmp9 = at::vec::Vectorized<float>(static_cast<float>(0.0));
            auto tmp10 = to_float_mask(tmp8 <= tmp9);
            auto tmp11 = tmp0.neg();
            auto tmp12 = decltype(tmp9)::blendv(tmp11, tmp9, tmp10);
            auto tmp13 = tmp7 + tmp12;
            tmp13.store(out_ptr0 + static_cast<long>(i0));
        }
    }
}
''')
```

C++, OpenMP, CUDA, Triton

Backend Graph Optimization



Operators Fusion, Memory Planning,...