

What is Regional Inductor?

torch.compile normally maps each captured graph to one Inductor compilation. Regional Inductor lets you tag FX operations for Inductor lowering while the unmarked region stays in the traced ATen graph.

Full-graph vs regional lowering

FULL GRAPH



REGIONAL



Illustrative FX-like graph (simplified)

```
def forward(self, x):
    a = torch.ops.aten.sin.default(x)
    b = compiled_inductor_region_0(a)
    c = torch.ops.aten.add.Tensor(b, x)
    d = compiled_inductor_region_1(c)
    return torch.ops.aten.cos.default(d)
```

Why regional lowering?

- Graph passes** Keep unlowered ATen nodes visible to graph transforms.
- Numerical scope** Lower only regions whose numerical equivalence was verified.
- Compile friction** Avoid slow full-graph lowering or failures in unselected region.

regional compile vs regional inductor

If the surrounding code do not need to remain in the same traced ATen graph, you can also just torch.compile the selected code region directly. Example: [Hugging Face Diffusers does this for its UNet](#): `pipe.unet = torch.compile(pipe.unet)`.

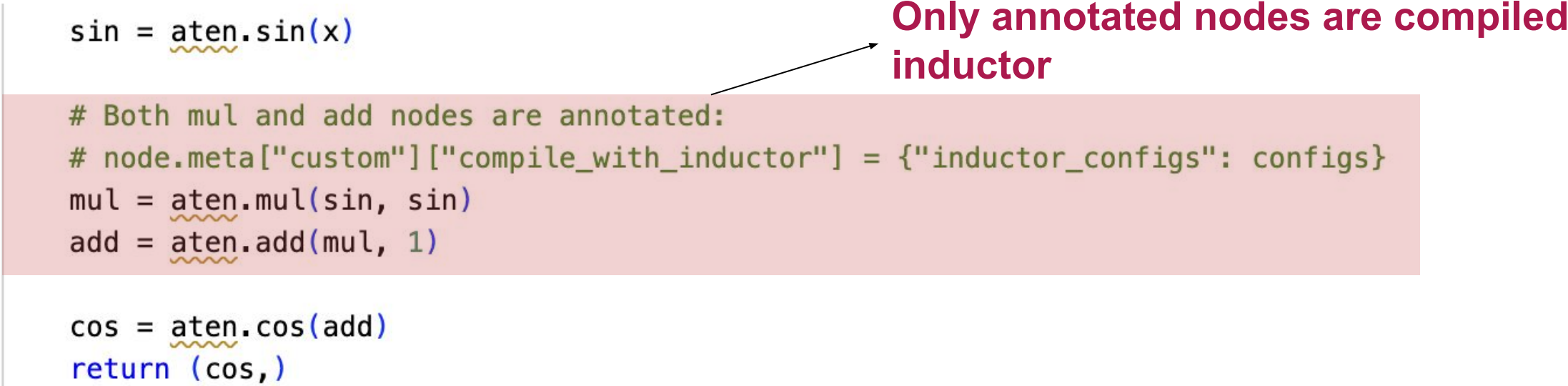
API for Regional Inductor – torch.fx.traceback.annotate

A context manager, tag FX operations under its context for selective lowering, untagged ops stay ATen.

```
from torch.fx.passes.regional_inductor import regional_inductor
annotate_backend = aot_autograd(fw_compiler=regional_inductor)

@torch.compile(backend=annotate_backend, fullgraph=True)
def annotate_model(x):
    x = x.sin()
    with annotate({"compile_with_inductor": True}):
        x = x * x + 1
    return x.cos()
```

Traced FX Graph



API for Regional Inductor – torch.compiler.nested_compile_region(options)

EXPERIMENTAL

nested_compile_region marks a function as a reusable compilation region. Within `torch.compile`, it is captured as a separate subgraph, allowing compatible calls to share compiled code and reduce compilation time.

The experimental `options` argument lets you specify Inductor settings for the subgraph independently of the outer compilation's settings.

```
# Experimental selective lowering with nested_compile_region
from torch.fx.passes.regional_inductor_invoke_subgraph import (
    regional_inductor_invoke_subgraph,
)
from torch_higher_order_ops.invoke_subgraph import get_invoke_subgraph_compile_options
nested_backend = aot_autograd(fw_compiler=regional_inductor_invoke_subgraph)
# can configure fw/bw compiler, decomposition table, etc.
options = get_invoke_subgraph_compile_options()

with torch_dynamo.config.patch(enable_invoke_subgraph_regional_compile=True):

    @torch.compiler.nested_compile_region(options=options)
    def region(x):
        return x * x + 1

    @torch.compile(backend=nested_backend, fullgraph=True)
    def nested_model(x):
        y = region(x)
        z = region(y)
        return z.cos()

    nested_out = nested_model(x)
```

Traced FX Graph

