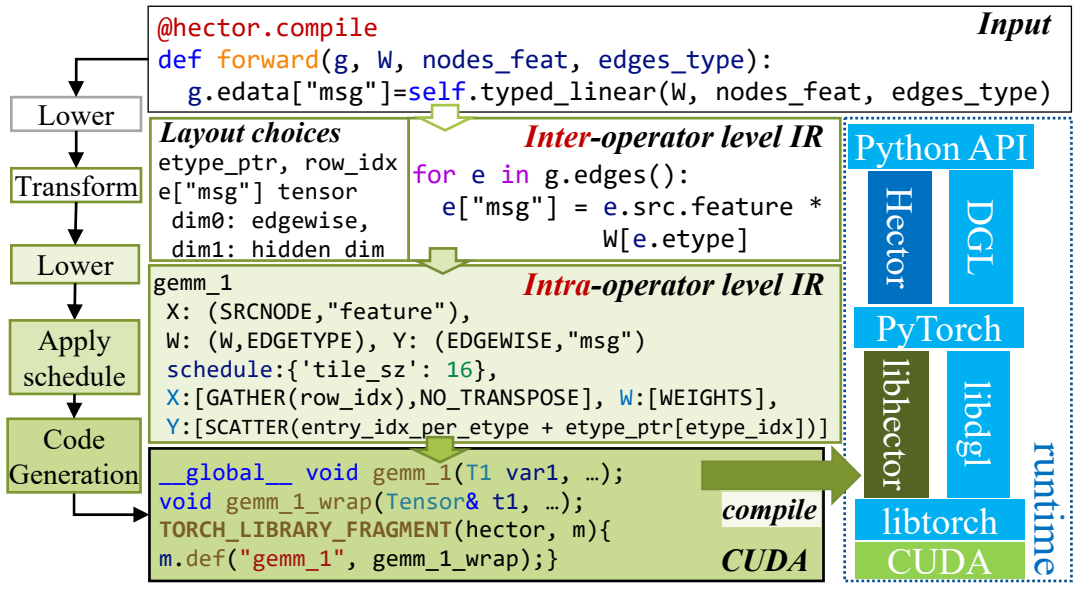
Input
@hector.compile
def forward(g, W, nodes_feat, edges_type):
g.edata["msg"]=self.typed_linear(W, nodes_feat, edges_type)
Lower
Transform
Lower
Apply schedule
Code Generation
Layout choices
etype_ptr, row_idx
e["msg"] tensor
dim0: edgewise,
dim1: hidden dim
Inter-operator level IR
for e in g.edges():
e["msg"] = e.src.feature *
W[e.etype]
Intra-operator level IR
gemm_1
X: (SRCNODE,"feature"),
W: (W,EDGETYPE), Y: (EDGEWISE,"msg")
schedule:{'tile_sz': 16},
X:[GATHER(row_idx),NO_TRANSPOSE], W:[WEIGHTS],
Y:[SCATTER(entry_idx_per_etype + etype_ptr[etype_idx])]
__global__ void gemm_1(T1 var1, …);
void gemm_1_wrap(Tensor& t1, …);
TORCH_LIBRARY_FRAGMENT(hector, m){
m.def("gemm_1", gemm_1_wrap);}
compile
CUDA
Python API
Hector
DGL
PyTorch
libhector
libdgl
libtorch
CUDA
runtime