

Defined Graph

X = Placeholder(X_meta)
W = Parameter(W_meta)
X = Gelu(X)
W = Reshard(W) # id=1
Y = Dot(X, W)
Y = Reshard(Y) # id=2

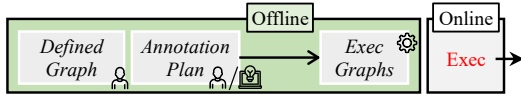
Annotation Plan

Leaf Operator's Output:
Placeholder(id=1): Annot_1
Parameter(id=1): Annot_2
Reshard Operator's Output:
Reshard(id=1): Annot_3
Reshard(id=2): Annot_4

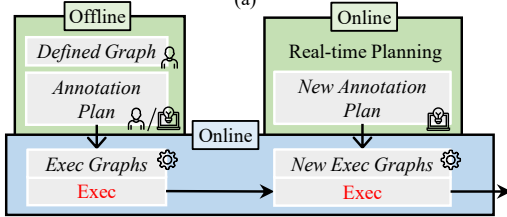
Exec Graphs

X = Placeholder(X_meta)
W = Parameter(W_meta)
X = Gelu(X)
W = SplitAG(W)
Y = Dot(X, W)
Y = RS(Y) GPU 0
...
Y = BSR() GPU 6

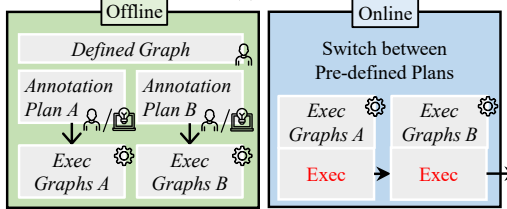
👤 : User 🗂️ : Planner
⚙️ : HSPMD System



(a)



(b)



(c)

🟢 : Graph Specialization (§6) 🟡 : Graph Switching (§7)

The program runs in an SPMD fashion
Each device launches and executes the same code

```
0 # Offline Graph Specialization
1 EGraph = DGraph.spec(AnnotPlan, local_device) # Fixed Strategy
2 # Run EGraph (device-specific executable graph) w/o data
3 # This will only deploy and shard parameters & optimizer states
4 # Can be regarded as model initialization
5 EGraph.exec(None)
6 # Online Part
7 # No Reconfiguration
8 for data in dataset:
9     EGraph.exec(data) # Training Step
```

(a)

```
0 # Offline Spec. (same as scenario (a))
1 EGraph = DGraph.spec(AnnotPlan, local_device) # Initial Strategy
2 EGraph.exec(None) # Init Model
3 for data in dataset:
4     if elastic_check(): # Trigger Reconfiguration
5         NewP = Planner.plan() # New Strategy
6         OldEGraph = EGraph
7         EGraph = DGraph.spec(NewP, local_device) # Online Spec.
8         DGraph.switch(OldEGraph, EGraph) # Online Switching
9     EGraph.exec(data) # Training Step
```

(b)

```
0 AnnotPlans = Planner.plan_all(dataset) # Prepare All Strategies
1 P2G = {P: DGraph.spec(P, local_device) for P in AnnotPlans}
2 EGraph = P2G.values[0] # Use First Parallel Strategy as Default
3 EGraph.exec(None) # Init Model
4 for data in dataset:
5     P = Planner.select(AnnotPlans, data) # Strategy Selection
6     if P2G[P] != EGraph: # Trigger Reconfiguration
7         DGraph.switch(EGraph, P2G[P]) # Online Graph Switching
8         EGraph = P2G[P]
9     EGraph.exec(data) # Training Step
```

(c)