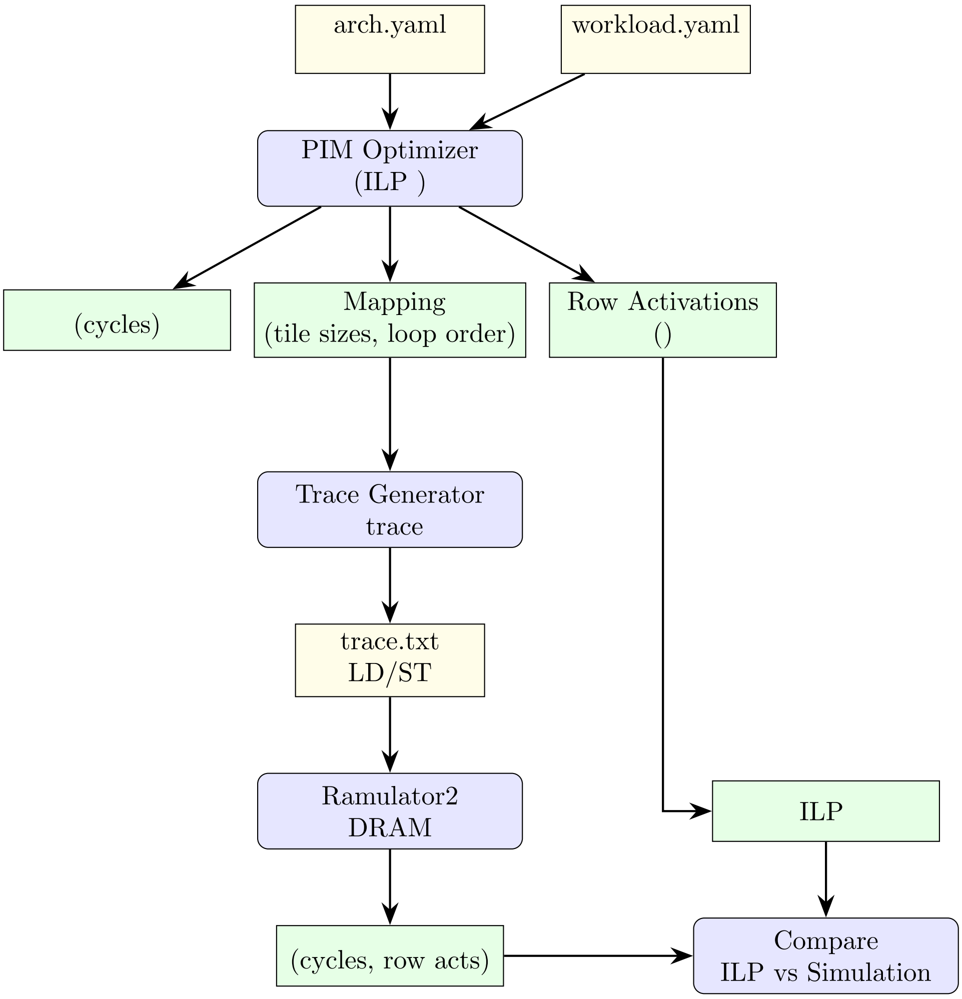
# PIM Optimizer

PIM Optimizer

December 23, 2025

# 1



# 2

## 2.1 1. PIM Optimizer (ILP )

- `arch.yaml`: PE array DRAM
- `workload`: (N, K, C, P, Q, R, S, stride, dilation)
- Tile sizes:
- Loop permutation:
- Row Activations

## 2.2 2. DRAM Latency Model

DRAM

$$\text{DRAM\_latency}_t = \frac{\text{mem\_reads}_t}{\text{BW}_{\text{rowbuf}}} + \text{row\_acts}_t \times T_{\text{act}} \tag{1}$$

DRAM

$$\text{DRAM\_latency}_{\text{total}} = \max(\text{DRAM\_latency}_{\text{input}}, \text{DRAM\_latency}_{\text{weight}}, \text{DRAM\_latency}_{\text{output}}) \tag{2}$$

### 2.3 3.

1. `print_mapping.py` - ILP mapping

↓

2. `trace_generator.py` - mapping trace

↓

3. `ramulator_runner.py` - Ramulator2

↓

4. `full_validation.py` - ILP

# 3

```
pim_optimizer/
 src/pim_optimizer/
    optimizer.py          #
    model/
       variables.py      # ILP
       constraints.py    # ILP
       objective.py      #  ( DRAM latency)
       row_activation.py # Row Activation
    arch/                 #
    workload/             #

 validation/dram/
    print_mapping.py      #  mapping
    trace_generator.py    #  Ramulator trace
    ramulator_runner.py   # Ramulator2
    full_validation.py    #

 examples/configs/
    arch.yaml             #
    conv_workload.yaml    #

 docs/
     dram_latency_model.tex  # DRAM
```

# 4

## 4.1

```
# Python API
from pim_optimizer import PIMOptimizer
from pim_optimizer.arch.pim_arch import PIMArchitecture
from pim_optimizer.workload.conv import ConvWorkload

arch = PIMArchitecture.from_yaml('examples/configs/arch.yaml')
workload = ConvWorkload(N=1, K=64, C=64, P=14, Q=14, R=3, S=3)
optimizer = PIMOptimizer(arch)
result = optimizer.optimize([workload])
```

### 4.2

```
# 1.  mapping
python validation/dram/print_mapping.py

# 2.  ( Ramulator2)
python validation/dram/full_validation.py
```