IPU HANDS-ON SESSION

Minkyu Kim
Field AI Engineer

GRAPHCORE

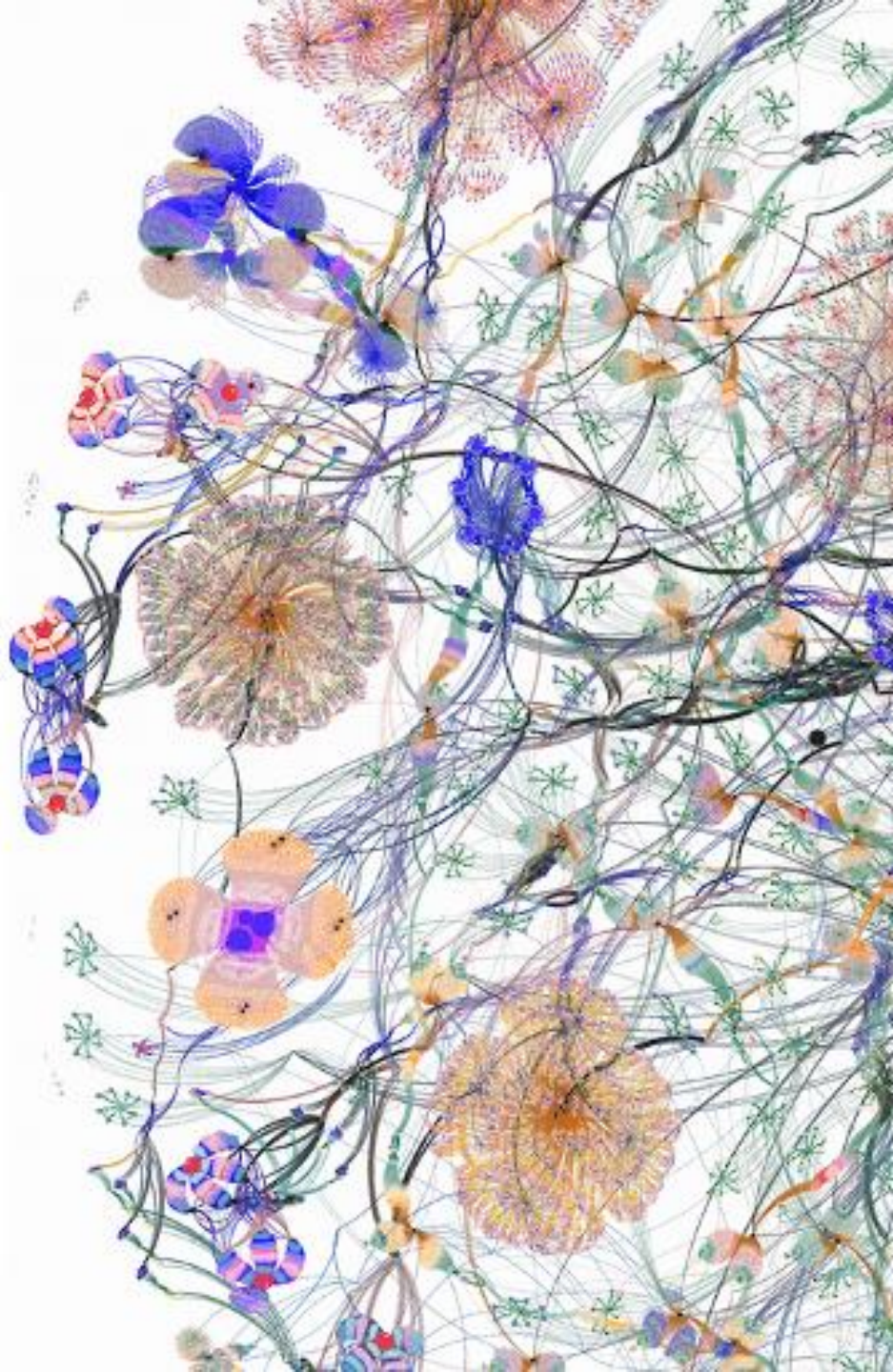
Reinforcement Learning Neural network visualization from POPLAR™

BERT LARGE – POPLAR GRAPH VISUALIZATION FROM POPART

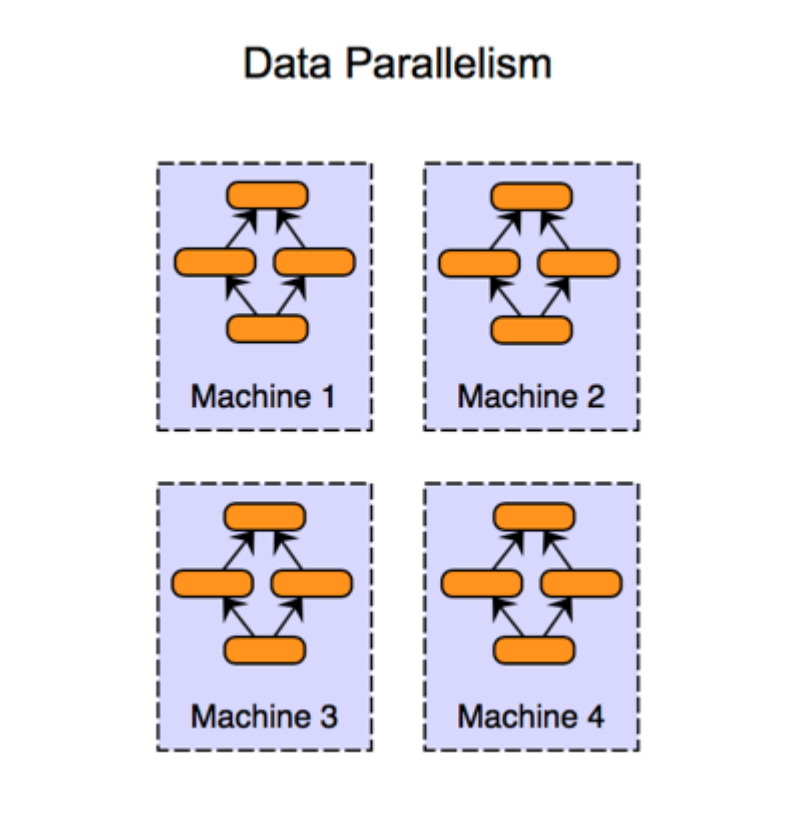
Day02

1. Multi IPU Programming
    - Data parallel
    - Model parallel
2. PopVision

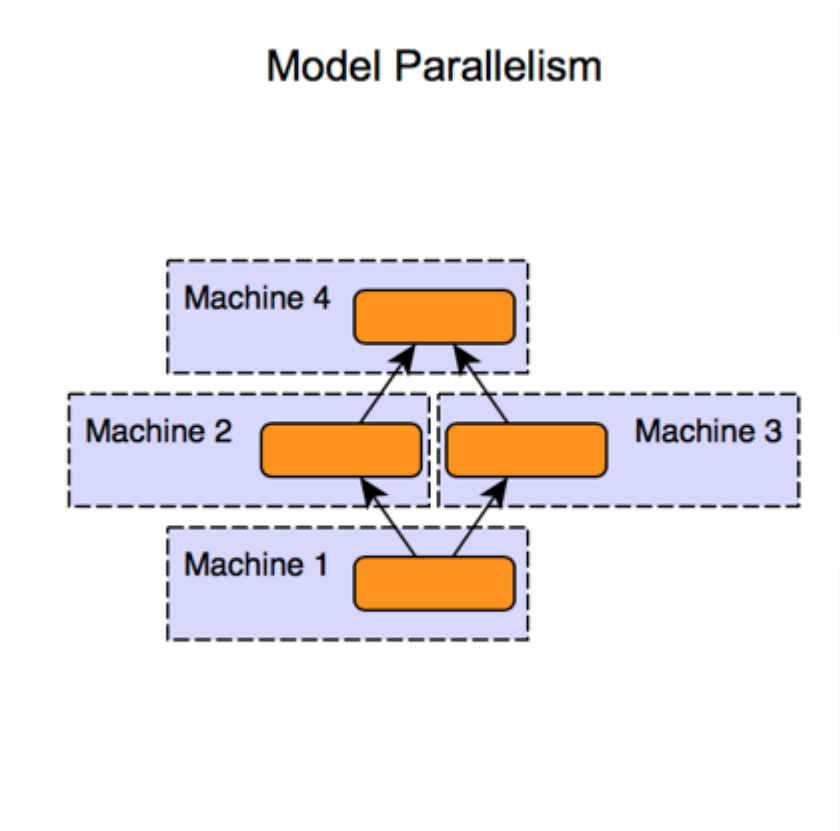
MULTI-IPU PROGRAMMING



DATA AND MODEL PARALLELISM



모델을 여러 칩에 복제하고 서로 다른 데이터를 input



하나의 모델을 여러 칩에 나누어 적재

DATA PARALLELISM

- 데이터 병렬처리

```
# Create a poptorch.Options instance to override default options
opts = poptorch.Options()

# Run a 100 iteration loop on the IPU, fetching a new batch each time
opts.deviceIterations(100)

# Duplicate the model over 4 replicas.
opts.replicationFactor(4)

training_data = poptorch.DataLoader(opts,
                                    dataset=ExampleDataset(shape=[3, 2],
                                                           length=100000),
                                    batch_size=model_batch_size,
                                    shuffle=True,
                                    drop_last=True)

model = ExampleModelWithLoss(data_shape=[3, 2], num_classes=2)
# Wrap the model in a PopTorch training wrapper
poptorch_model = poptorch.trainingModel(model, options=opts)
```

`replicationFactor(replication_factor)`

Number of times to replicate the model (default: 1).

Replicating the model increases the data throughput of the model as PopTorch uses more IPUs. This leads to the number of IPUs used being scaled by `replication_factor`, for example, if your model uses 1 IPU, a `replication_factor` of 2 will use 2 IPUs; if your model uses 4 IPUs, a replication factor of 4 will use 16 IPUs in total.

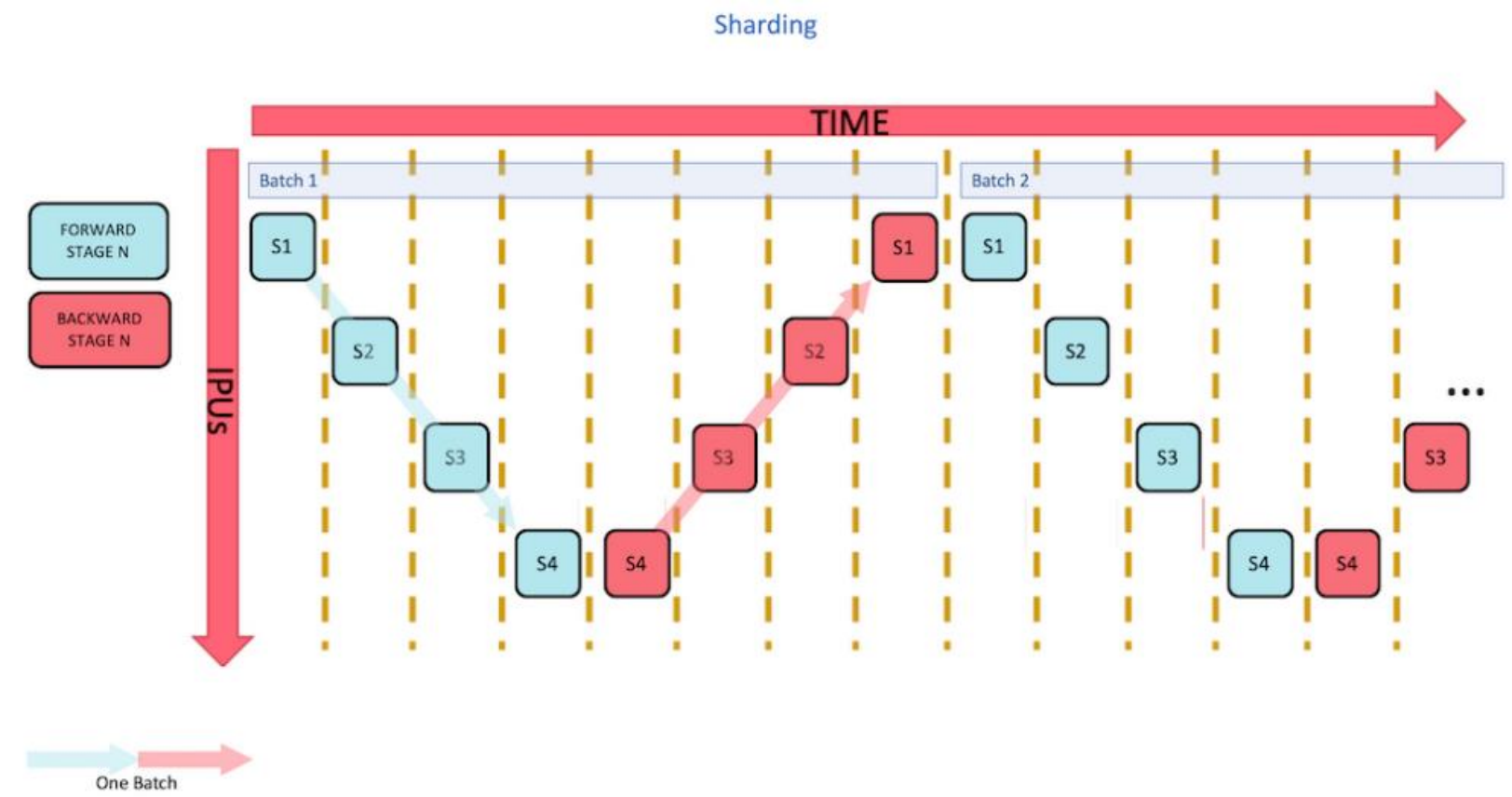
Parameters: **replication_factor** (*int*) – Number of replicas of the model to create.

Return type: poptorch.Options

https://docs.graphcore.ai/projects/tensorflow-user-guide/en/latest/tensorflow/keras_tf2.html#automatic-data-parallelism

MODEL PARALLELISM

- Model Sharding



https://github.com/graphcore/tutorials/tree/master/tutorials/tensorflow1/pipelining

MODEL PARALLELISM

- Model Sharding

GPU pytorch 예시 코드

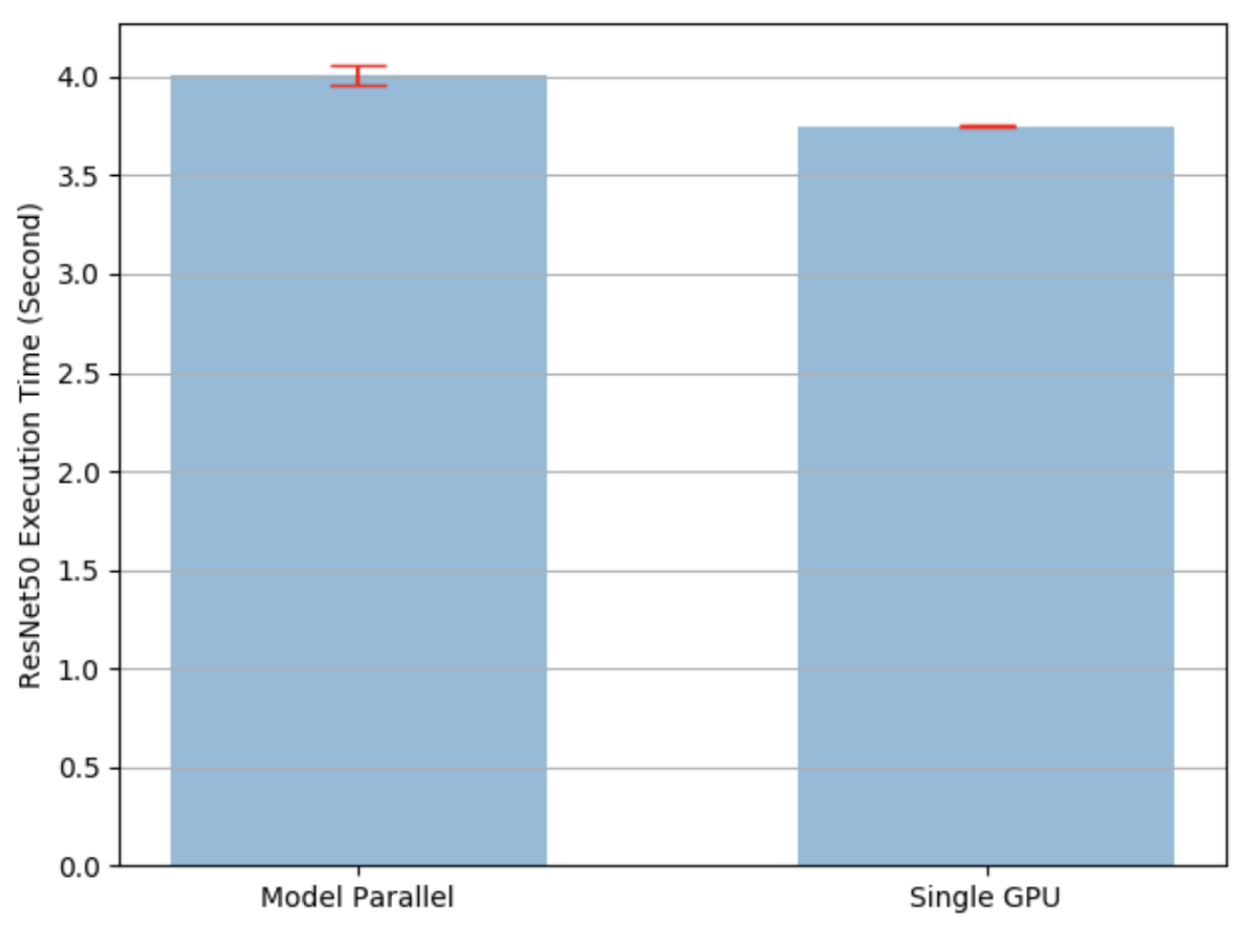
```python
from torchvision.models.resnet import ResNet, Bottleneck

num_classes = 1000


class ModelParallelResNet50(ResNet):
    def __init__(self, *args, **kwargs):
        super(ModelParallelResNet50, self).__init__(
            Bottleneck, [3, 4, 6, 3], num_classes=num_classes, *args, **kwargs)

        self.seq1 = nn.Sequential(
            self.conv1,
            self.bn1,
            self.relu,
            self.maxpool,

            self.layer1,
            self.layer2
        ).to('cuda:0')

        self.seq2 = nn.Sequential(
            self.layer3,
            self.layer4,
            self.avgpool,
        ).to('cuda:1')

        self.fc.to('cuda:1')

    def forward(self, x):
        x = self.seq2(self.seq1(x).to('cuda:1'))
        return self.fc(x.view(x.size(0), -1))
```


모델을 GPU0와 GPU1에 나누어 적재

https://pytorch.org/tutorials/intermediate/model_parallel_tutorial.html#single-machine-model-parallel-best-practices

MODEL PARALLELISM

- Model Sharding

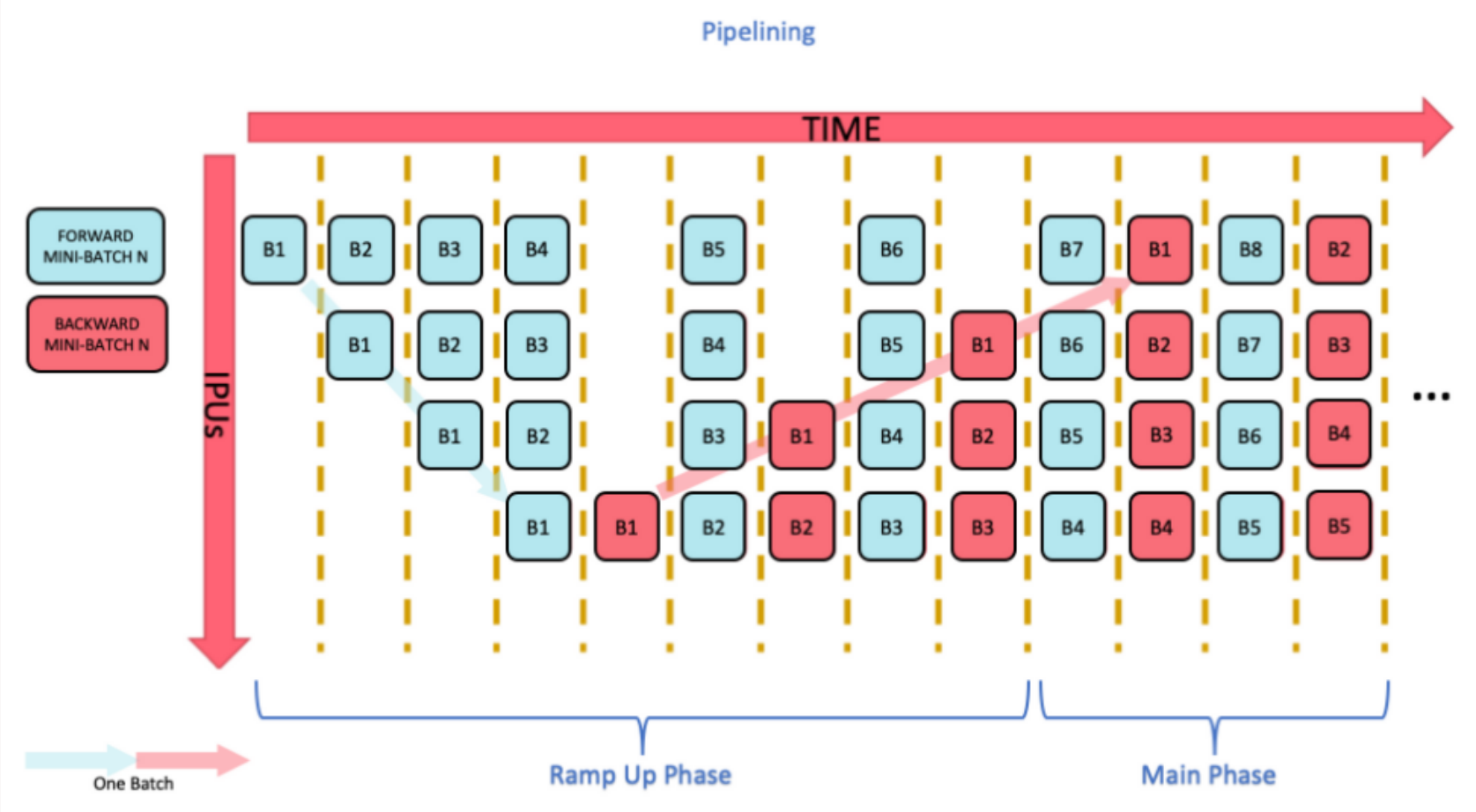


=> Single GPU 보다 느림

https://pytorch.org/tutorials/intermediate/model_parallel_tutorial.html#single-machine-model-parallel-best-practices

MODEL PARALLELISM

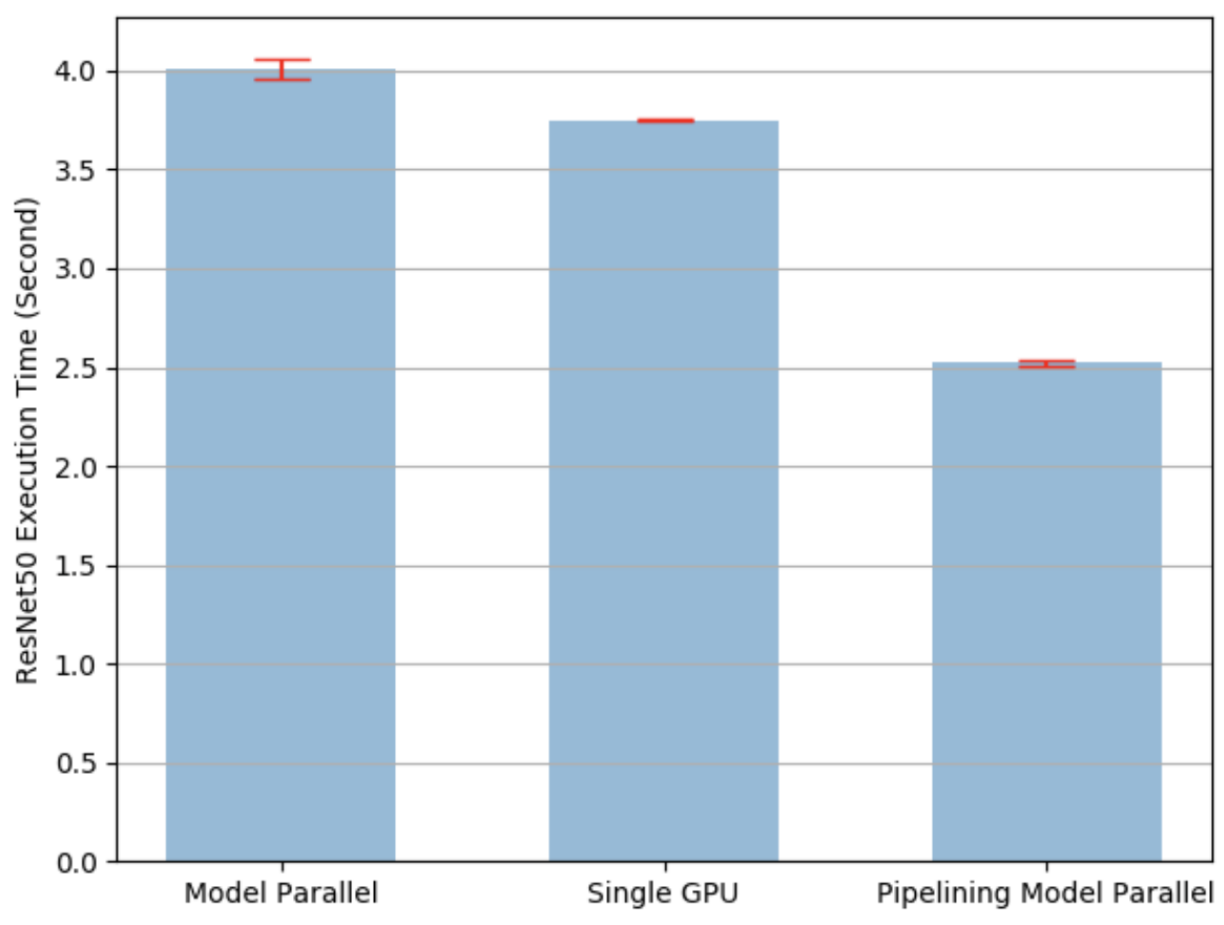
- Model Pipelining



https://github.com/graphcore/tutorials/tree/master/tutorials/tensorflow1/pipelining

MODEL PARALLELISM

- Model Pipelining



=> Pipelining 속도 향상

https://github.com/graphcore/tutorials/tree/master/tutorials/tensorflow1/pipelining

MODEL PARALLELISM

- Model Pipelining

GPU pytorch 예시 코드

```
class PipelineParallelResNet50(ModelParallelResNet50):
    def __init__(self, split_size=20, *args, **kwargs):
        super(PipelineParallelResNet50, self).__init__(*args, **kwargs)
        self.split_size = split_size

    def forward(self, x):
        splits = iter(x.split(self.split_size, dim=0))
        s_next = next(splits)
        s_prev = self.seq1(s_next).to('cuda:1')
        ret = []

        for s_next in splits:
            # A. s_prev runs on cuda:1
            s_prev = self.seq2(s_prev)
            ret.append(self.fc(s_prev.view(s_prev.size(0), -1)))

            # B. s_next runs on cuda:0, which can run concurrently with A
            s_prev = self.seq1(s_next).to('cuda:1')

        s_prev = self.seq2(s_prev)
        ret.append(self.fc(s_prev.view(s_prev.size(0), -1)))

        return torch.cat(ret)
```


iterator와 for문을 사용하여 pipelining 직접 구현

https://pytorch.org/tutorials/intermediate/model_parallel_tutorial.html#single-machine-model-parallel-best-practices

MODEL PARALLELISM

- IPU의 Model Pipelining

IPU의 Poplar SDK에서는 API를 통해 split만 나누어주면 pipelining 자동 실행

1) 모델 Layer를 직접 작성하는 경우

```
class Network(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.layer1 = torch.nn.Linear(5, 10)
        self.layer2 = torch.nn.Linear(10, 5)
        self.layer3 = torch.nn.Linear(5, 5)
        self.layer4 = torch.nn.Linear(5, 5)

        self.act = torch.nn.ReLU()
        self.softmax = torch.nn.Softmax(dim=1)
```

```
def forward(self, x):

    # Explicit layers on a certain IPU
    poptorch.Block.useAutoId()
    with poptorch.Block(ipu_id=0):
        x = self.act(self.layer1(x))

    with poptorch.Block(ipu_id=1):
        x = self.act(self.layer2(x))

    with poptorch.Block(ipu_id=2):
        x = self.act(self.layer3(x))
        x = self.act(self.layer4(x))

    with poptorch.Block(ipu_id=3):
        x = self.softmax(x)
    return x
```

Layer1 → IPU0

Layer2 → IPU1

Layer3, 4 → IPU2

Softmax → IPU3

https://docs.graphcore.ai/projects/poptorch-user-guide/en/2.0.0/overview.html#annotation-tools

MODEL PARALLELISM

- IPU의 Model Pipelining

IPU의 Poplar SDK에서는 API를 통해 split만 나누어주면 pipelining 자동 실행

2) 모델을 import하여 사용하는 경우

```
class WrappedModel(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.wrapped = transformers.BertForQuestionAnswering.from_pretrained(
            pretrained_weights)
```

```
# All layers before "model.bert.encoder.layer[0]" will be on IPU 0 and all layers from
# "model.bert.encoder.layer[0]" onwards (inclusive) will be on IPU 1.
model.bert.encoder.layer[0] = poptorch.BeginBlock(model.bert.encoder.layer[0],
                                                  ipu_id=1)

# Now all layers before layer are on IPU 1 and this layer onward is on IPU 2
model.bert.encoder.layer[2] = poptorch.BeginBlock(model.bert.encoder.layer[2],
                                                  ipu_id=2)

# Finally all layers from this layer till the end of the network are on IPU 3.
model.bert.encoder.layer[4] = poptorch.BeginBlock(model.bert.encoder.layer[4],
                                                  ipu_id=3)
```

기본으로 전체 → IPU0

Layer0 부터 → IPU1

Layer2 부터 → IPU2

Layer4 부터 → IPU3

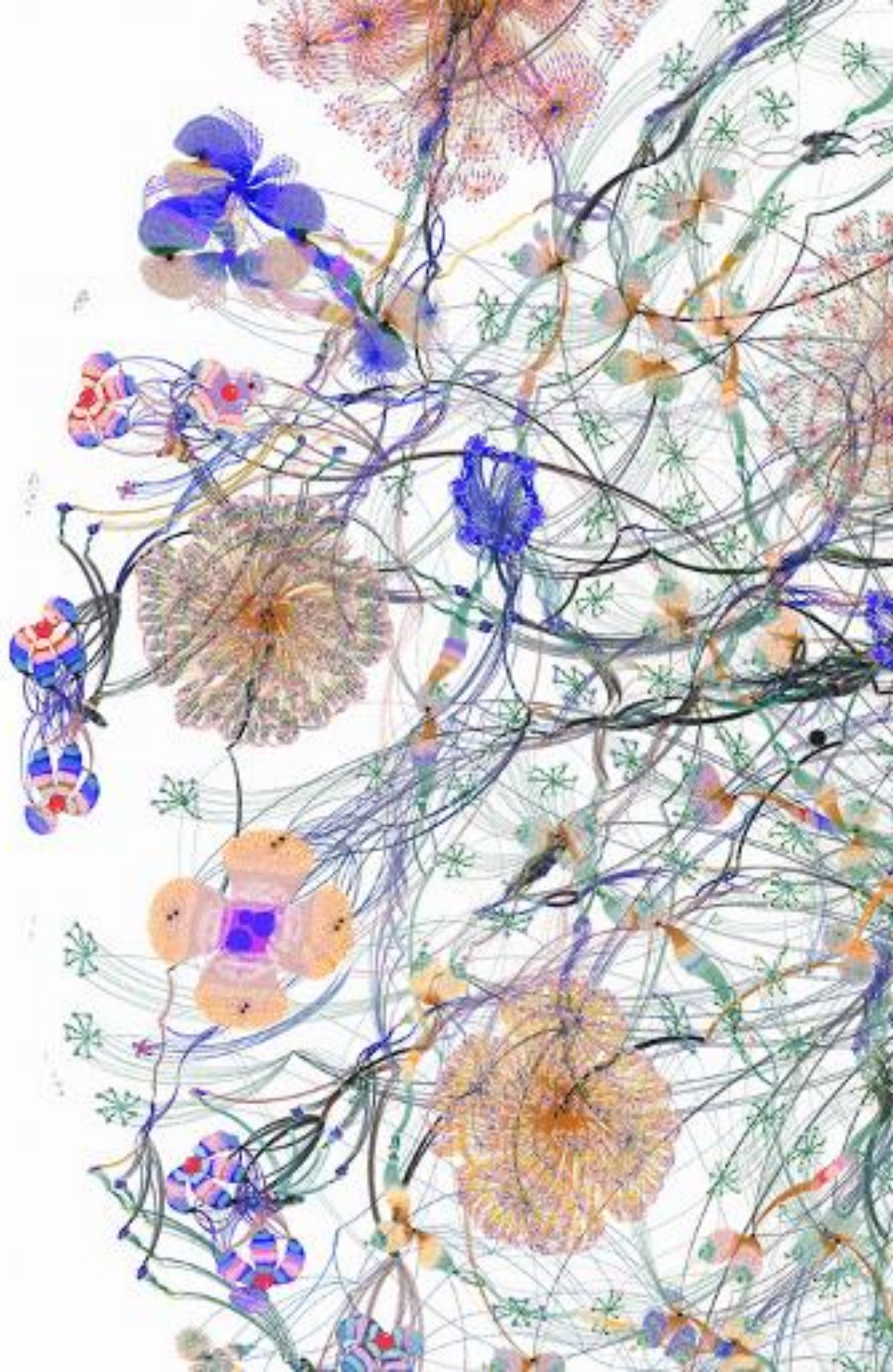
https://docs.graphcore.ai/projects/poptorch-user-guide/en/2.0.0/overview.html#annotation-tools

MODEL PARALLELISM

Pipelining Popvision 시각화

https://docs.graphcore.ai/projects/tf-model-parallelism/en/latest/pipelining.html

HANDS-ON TUTORIAL



GRAPHCORE COMMAND LINE TOOLS

- gc-monitor : 현재 파티션 내 ipu 하드웨어 정보 및 ipu 프로세스 확인

```
(venv_tutorial) minkyuk@pod642:~/projects/NHN/hands_on_tutorials/day01$ gc-monitor
+-------------+-----------------------------------------------------------------------------------------------------------+
| gc-monitor  |                          Partition: 'pt-minkyuk-16-ipus' has 16 reconfigurable IPUs                       |
+-------------+-------------------+----------+------+--------------+----+--------+-------+-----------------------------+
|    IPU-M    |       Serial      | ICU FW   | Type |Server version| ID | PCIe ID |Routing|             GWSW            |
+-------------+-------------------+----------+------+--------------+----+--------+-------+-----------------------------+
| 10.1.5.21   | 0046.0002.8204521 | 2.4.4    |M2000 |    1.9.0     | 0  |    3   | DNC   |             2.4.2           |
| 10.1.5.21   | 0046.0002.8204521 | 2.4.4    |M2000 |    1.9.0     | 1  |    2   | DNC   |             2.4.2           |
| 10.1.5.21   | 0046.0001.8204521 | 2.4.4    |M2000 |    1.9.0     | 2  |    1   | DNC   |             2.4.2           |
| 10.1.5.21   | 0046.0001.8204521 | 2.4.4    |M2000 |    1.9.0     | 3  |    0   | DNC   |             2.4.2           |
+-------------+-------------------+----------+------+--------------+----+--------+-------+-----------------------------+
| 10.1.5.22   | 0012.0002.8204521 | 2.4.4    |M2000 |    1.9.0     | 4  |    3   | DNC   |             2.4.2           |
| 10.1.5.22   | 0012.0002.8204521 | 2.4.4    |M2000 |    1.9.0     | 5  |    2   | DNC   |             2.4.2           |
| 10.1.5.22   | 0012.0001.8204521 | 2.4.4    |M2000 |    1.9.0     | 6  |    1   | DNC   |             2.4.2           |
| 10.1.5.22   | 0012.0001.8204521 | 2.4.4    |M2000 |    1.9.0     | 7  |    0   | DNC   |             2.4.2           |
+-------------+-------------------+----------+------+--------------+----+--------+-------+-----------------------------+
| 10.1.5.23   | 0040.0002.8204521 | 2.4.4    |M2000 |    1.9.0     | 8  |    3   | DNC   |             2.4.2           |
| 10.1.5.23   | 0040.0002.8204521 | 2.4.4    |M2000 |    1.9.0     | 9  |    2   | DNC   |             2.4.2           |
| 10.1.5.23   | 0040.0001.8204521 | 2.4.4    |M2000 |    1.9.0     | 10 |    1   | DNC   |             2.4.2           |
| 10.1.5.23   | 0040.0001.8204521 | 2.4.4    |M2000 |    1.9.0     | 11 |    0   | DNC   |             2.4.2           |
+-------------+-------------------+----------+------+--------------+----+--------+-------+-----------------------------+
| 10.1.5.24   | 0050.0002.8204521 | 2.4.4    |M2000 |    1.9.0     | 12 |    3   | DNC   |             2.4.2           |
| 10.1.5.24   | 0050.0002.8204521 | 2.4.4    |M2000 |    1.9.0     | 13 |    2   | DNC   |             2.4.2           |
| 10.1.5.24   | 0050.0001.8204521 | 2.4.4    |M2000 |    1.9.0     | 14 |    1   | DNC   |             2.4.2           |
| 10.1.5.24   | 0050.0001.8204521 | 2.4.4    |M2000 |    1.9.0     | 15 |    0   | DNC   |             2.4.2           |
+-------------+-------------------+----------+------+--------------+----+--------+-------+-----------------------------+
+-----------------------------------------------------------------------------------------+
|                                  No attached processes                                  |
+-----------------------------------------------------------------------------------------+
```

- vipu-admin list partitions : IPU-POD 내 파티션 정보 확인

https://docs.graphcore.ai/projects/command-line-tools/en/latest/index.html

SETUP

1. Poplar SDK 활성화
   - source SDK_PATH/poplar*/enable.sh
   - source SDK_PATH/popart*/enable.sh

2. 가상환경 생성 및 활성화
   - virtualenv -p python3 venv_tutorial
   - source venv_tutorial/bin/activate

3. 패키지 설치
   - (pytorch)pip install SDK_PATH/poptorch*.whl
   - (tensorflow)pip install SDK_PATH/tensorflow-2*amd*.whl (tf1/tf2, amd/intel 지정)

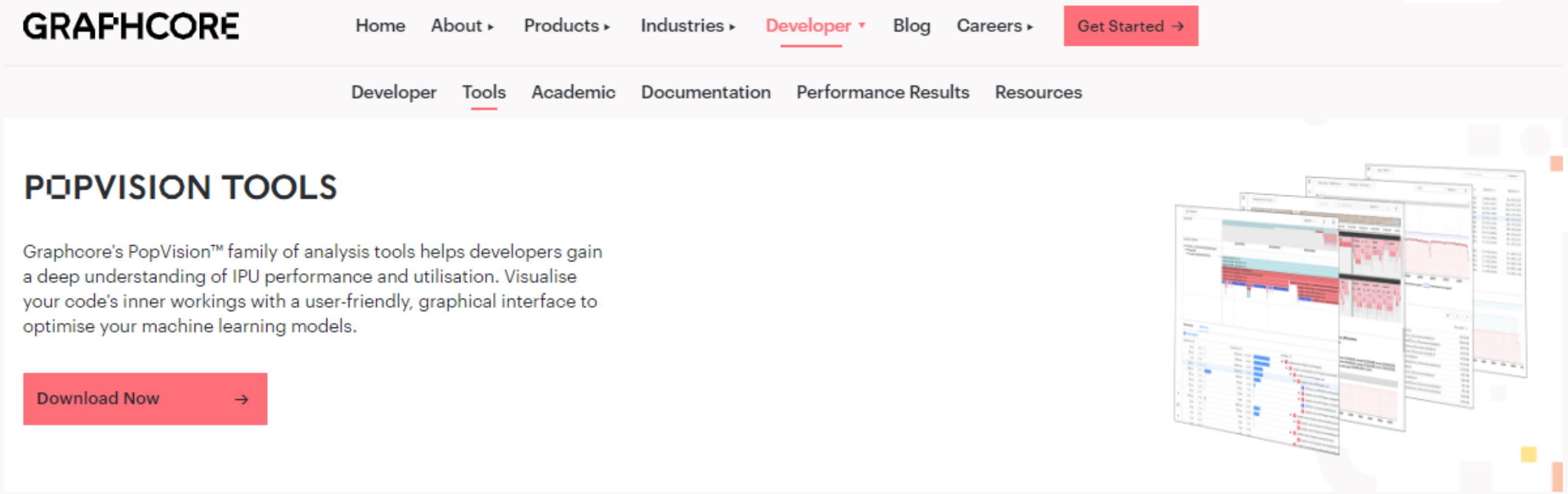
```
Package              Version
-------------------- -----------
dataclasses          0.8
importlib-resources  5.4.0
pip                  21.3.1
pkg_resources        0.0.0
poptorch             2.4.0+40669
setuptools           59.6.0
torch                1.10.0+cpu
tqdm                 4.64.0
typing_extensions    4.1.1
wheel                0.37.1
zipp                 3.6.0
```

poptorch 설치 시 호환되는 torch 버전 자동으로 함께 설치

```
(venv_tutorial) minkyuk@pod642:~/projects/NHN/hands_on_tutorials/day01$ ls ~/sdks/2.4.0/tensorflow-
tensorflow-1.15.5+gc2.4.0+139618+803bd5d97f3+amd_znver1-cp36-cp36m-linux_x86_64.whl       tensorflow-2.4.4+gc2.4.0+139613+8debb698097+amd_znver1-cp36-cp36m-linux_x86_64.whl
tensorflow-1.15.5+gc2.4.0+139620+803bd5d97f3+intel_skylake512-cp36-cp36m-linux_x86_64.whl  tensorflow-2.4.4+gc2.4.0+139619+8debb698097+intel_skylake512-cp36-cp36m-linux_x86_64.whl
```

TUTORIAL

Popvision Graph Analyzer



https://www.graphcore.ai/developer/popvision-tools

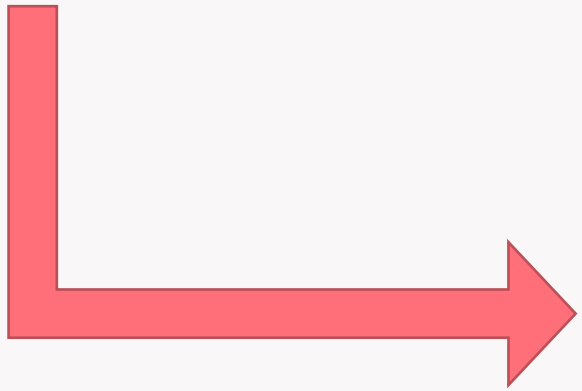
TUTORIAL

Popvision Graph Analyzer

```
POPLAR_ENGINE_OPTIONS='{"autoReport.all":"true", "autoReport.directory":"./DIR_NAME", "debug.allowOutOfMemory":"true"}'
```

예시

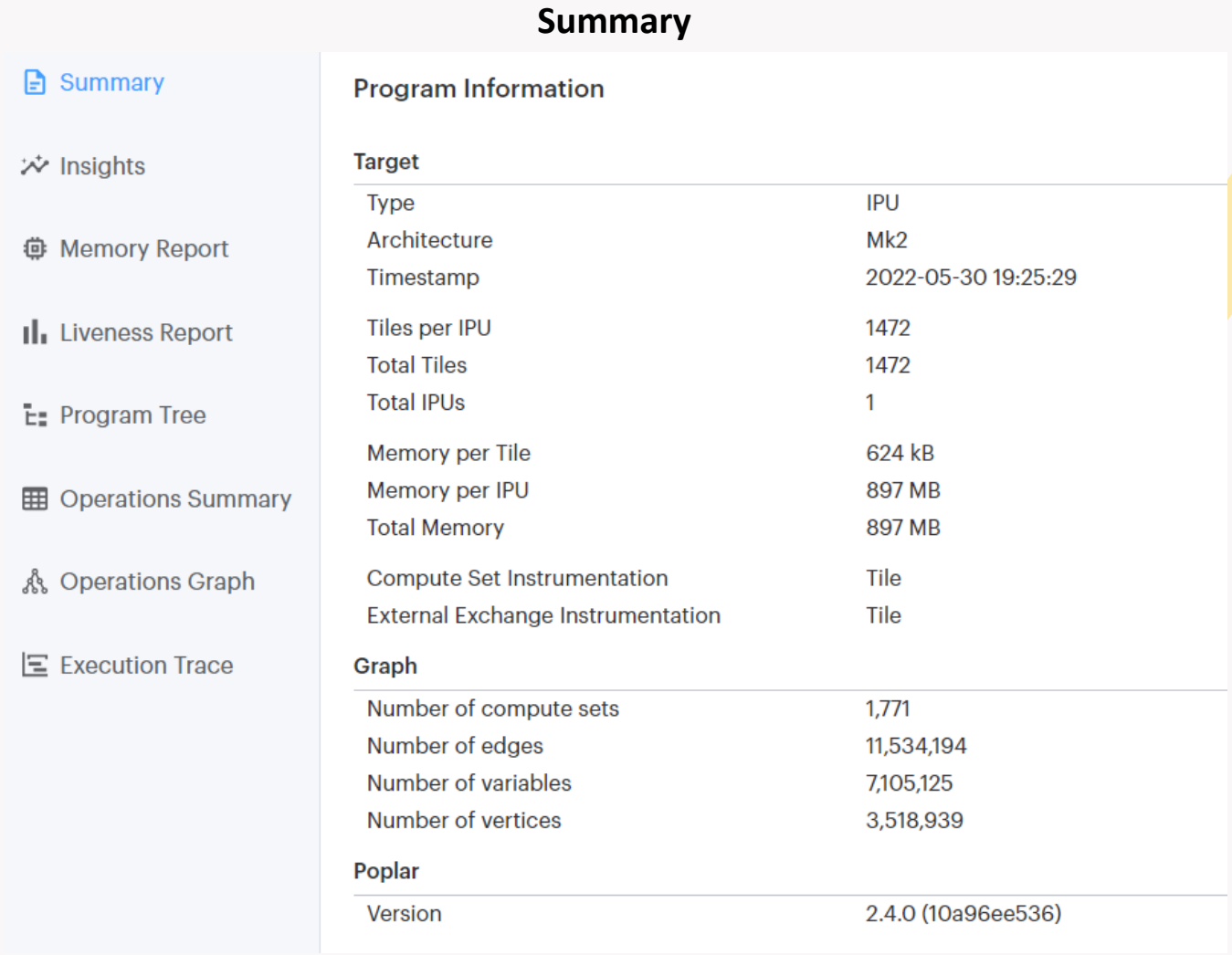
```
(venv_tutorial) minkyuk@pod642:~/projects/NHN/hands_on_tutorials/day02$ POPLAR_ENGINE_OPTIONS
='{"autoReport.all":"true", "autoReport.directory":"./reports/resnet50_model_data_parallel",
"debug.allowOutOfMemory":"true"}' python multi_ipu.py
```



```
(venv_tutorial) minkyuk@pod642:~/projects/NHN/hands_on_tutorials/day02$ tree reports/
reports/
├── resnet50
│   ├── debug.cbor
│   └── training
│       ├── archive.a
│       ├── debug.cbor -> ../debug.cbor
│       ├── profile.pop
│       └── profile.pop_cache
├── resnet50_model_data_parallel
│   ├── debug.cbor
│   └── training
│       ├── archive.a
│       ├── debug.cbor -> ../debug.cbor
│       └── profile.pop
└── resnet50_model_parallel
    ├── debug.cbor
    └── training
        ├── archive.a
        ├── debug.cbor -> ../debug.cbor
        ├── profile.pop
        └── profile.pop_cache
```

TUTORIAL

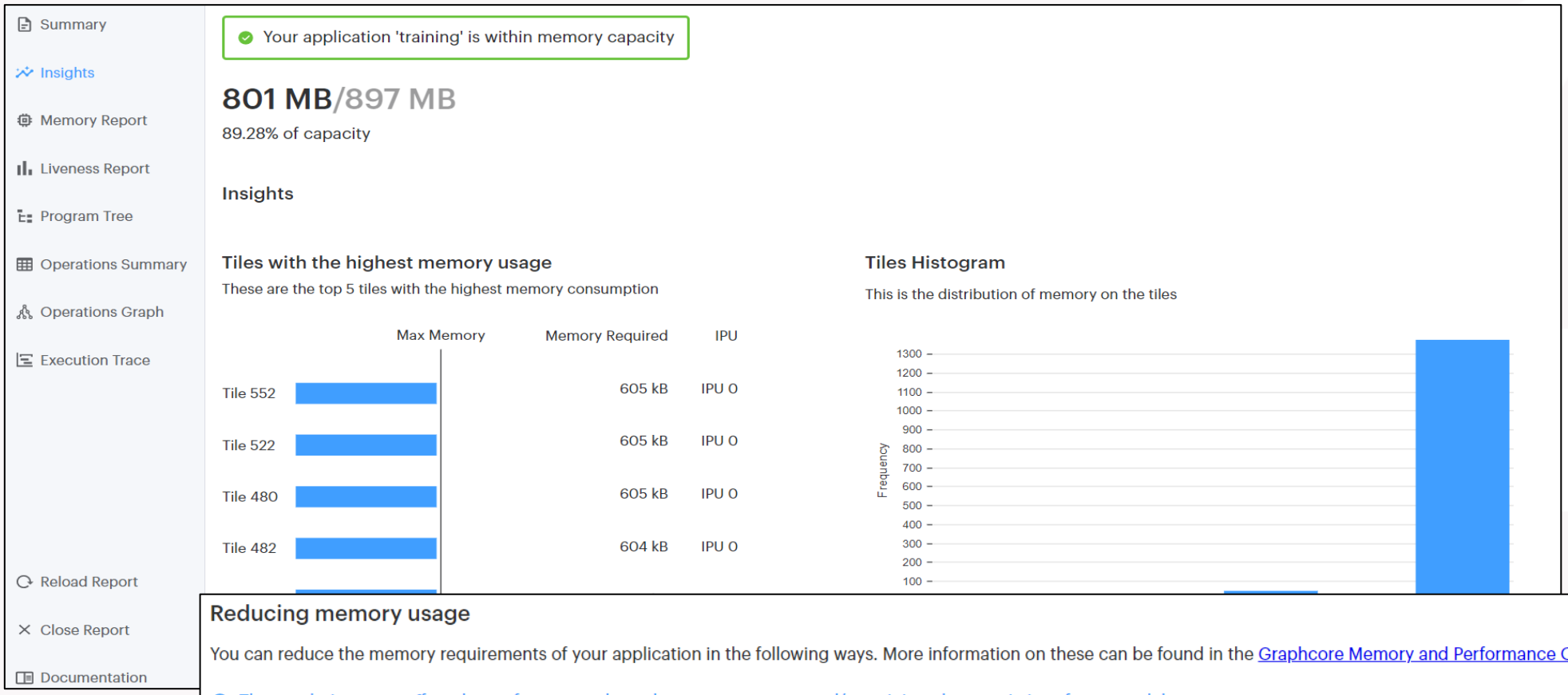
- base.py

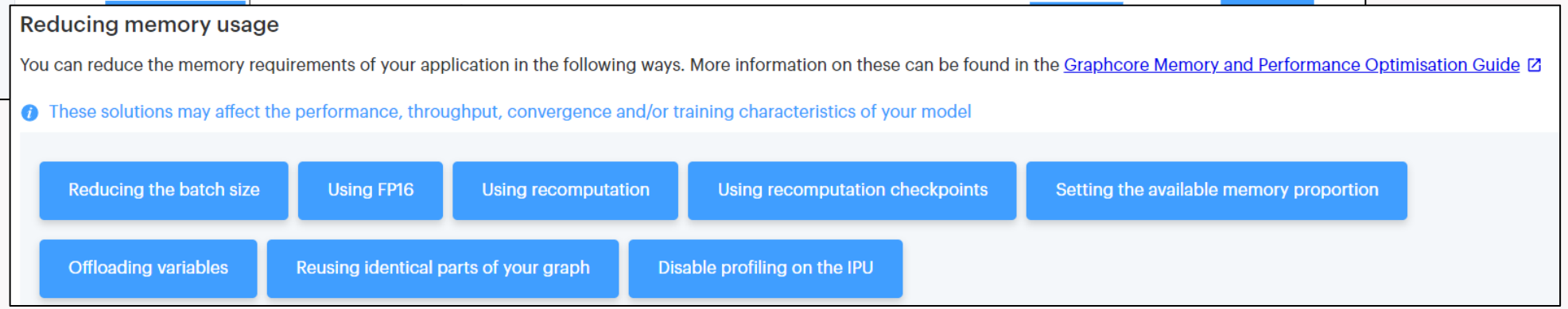


TUTORIAL

- base.py

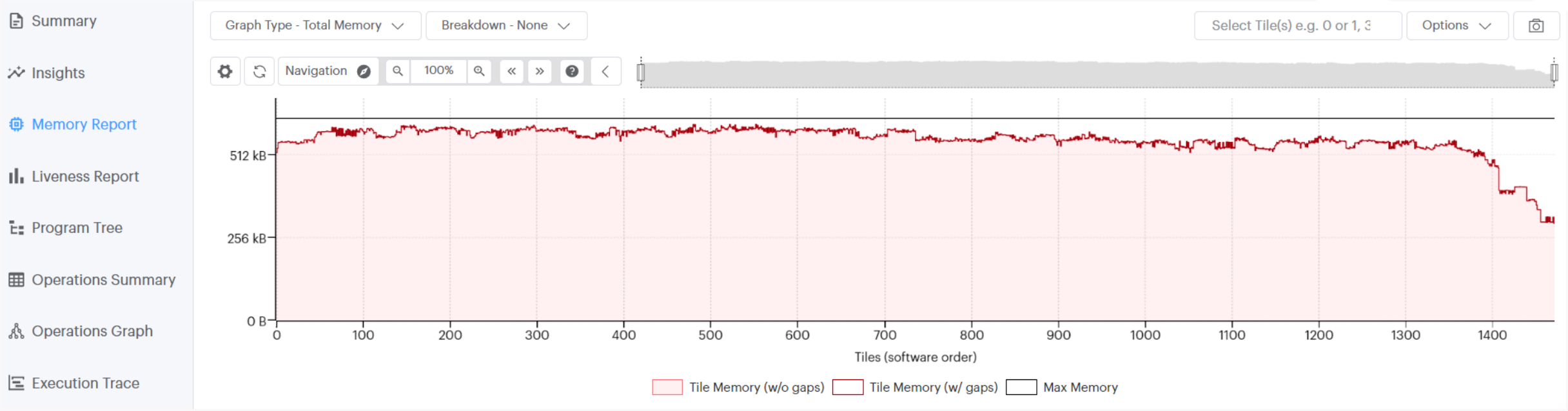
Insights





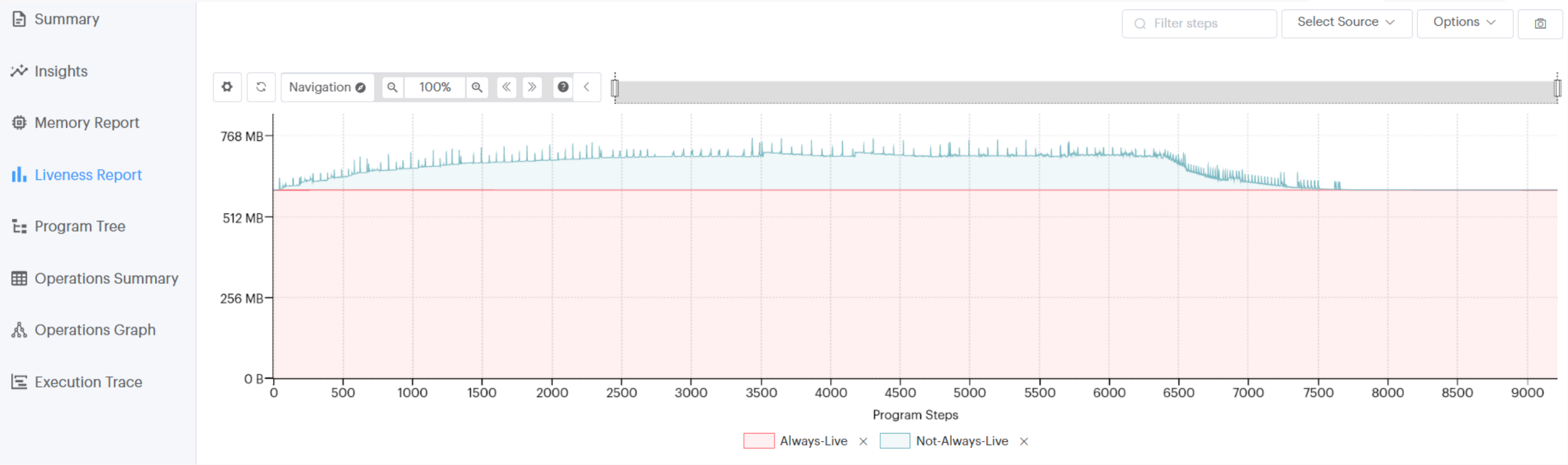
TUTORIAL

- base.py



TUTORIAL

- base.py

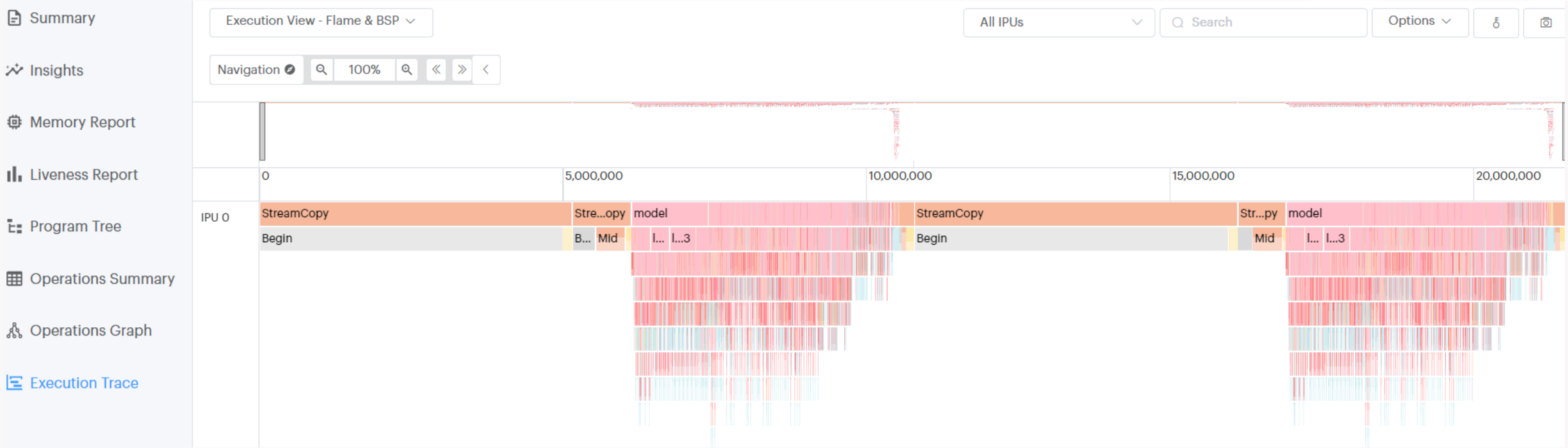


- Always-Live : 그래프 연산의 lifetime 전체에 걸쳐 메모리를 점유하는 변수. Ex) code, constants
- Not-Always-Live : 특정 프로그램 step에서만 사용되는 변수.

TUTORIAL

- base.py

Execution Trace



TUTORIAL

- Model Parallel

```
56    opts = poptorch.Options()
57
58    model = torchvision.models.resnet50()
```

```
57    opts = poptorch.Options()
58+   opts.Training.gradientAccumulation(3)
59
60    model = torchvision.models.resnet50()
61+   model.layer4 = poptorch.BeginBlock(model.layer4, ipu_id=1)
```

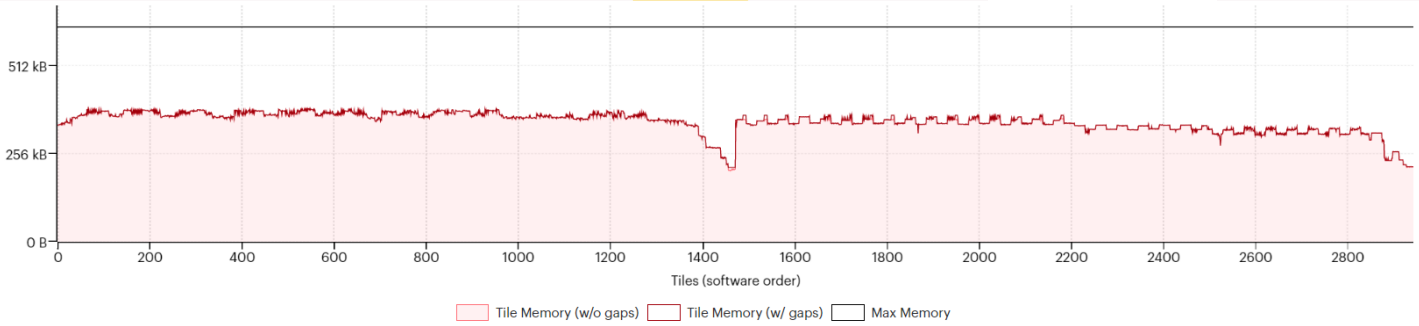
- Summary
- Insights
- Memory Report
- Liveness Report
- Program Tree
- Operations Summary
- Operations Graph
- Execution Trace

Program Information

Target	
Type	IPU
Architecture	Mk2
Timestamp	2022-05-30 19:18:02
Tiles per IPU	1472
Total Tiles	2944
Total IPUs	2
Memory per Tile	624 kB
Memory per IPU	897 MB
Total Memory	1,794 MB
Compute Set Instrumentation	Tile
External Exchange Instrumentation	Tile

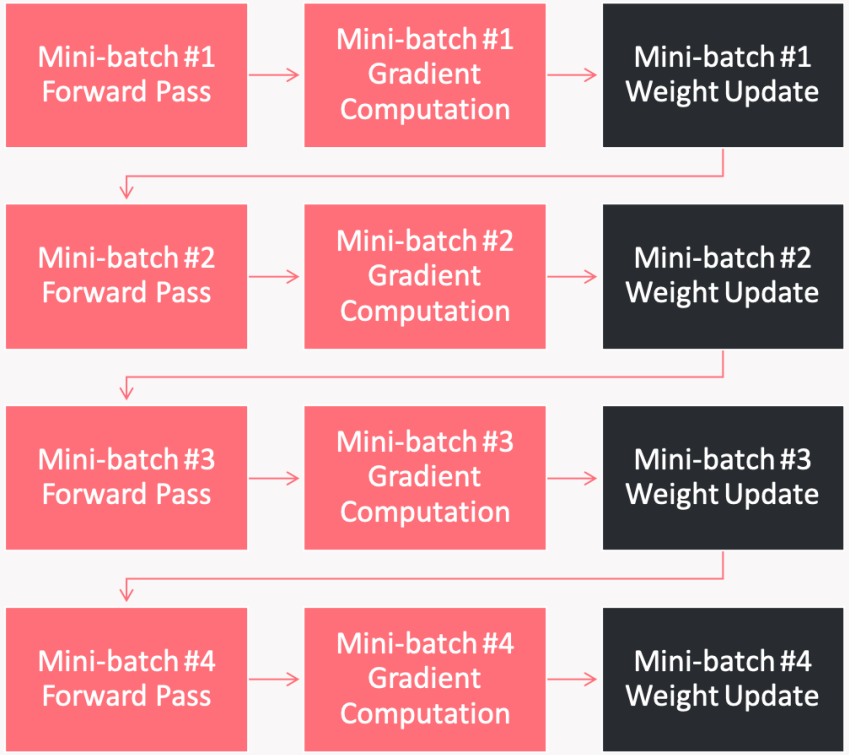
Graph	
Number of compute sets	1,994
Number of edges	12,866,597
Number of variables	7,857,014
Number of vertices	4,163,455

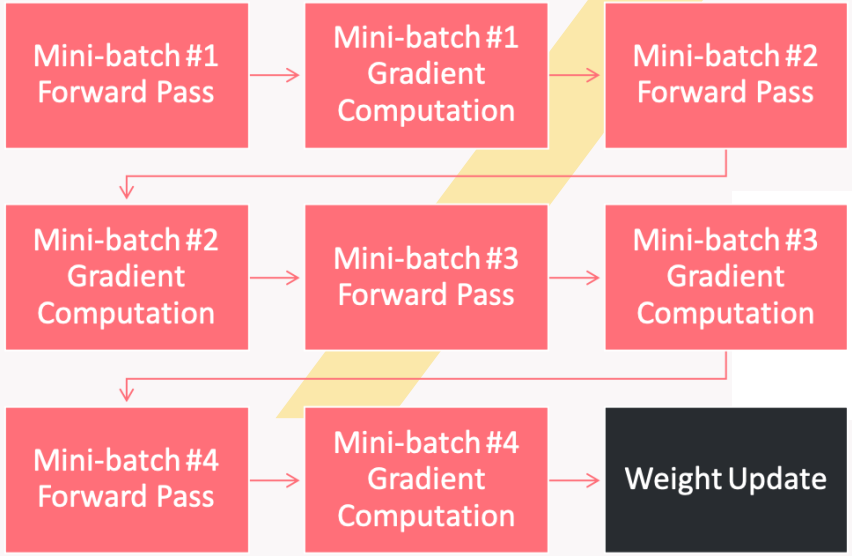
Poplar	
Version	2.4.0 (10a96ee536)



GRADIENT ACCUMULATION

- 한 번의 weight update에 더 많은 샘플 사용
- 메모리 사용은 유지하면서 배치사이즈 늘리는 효과

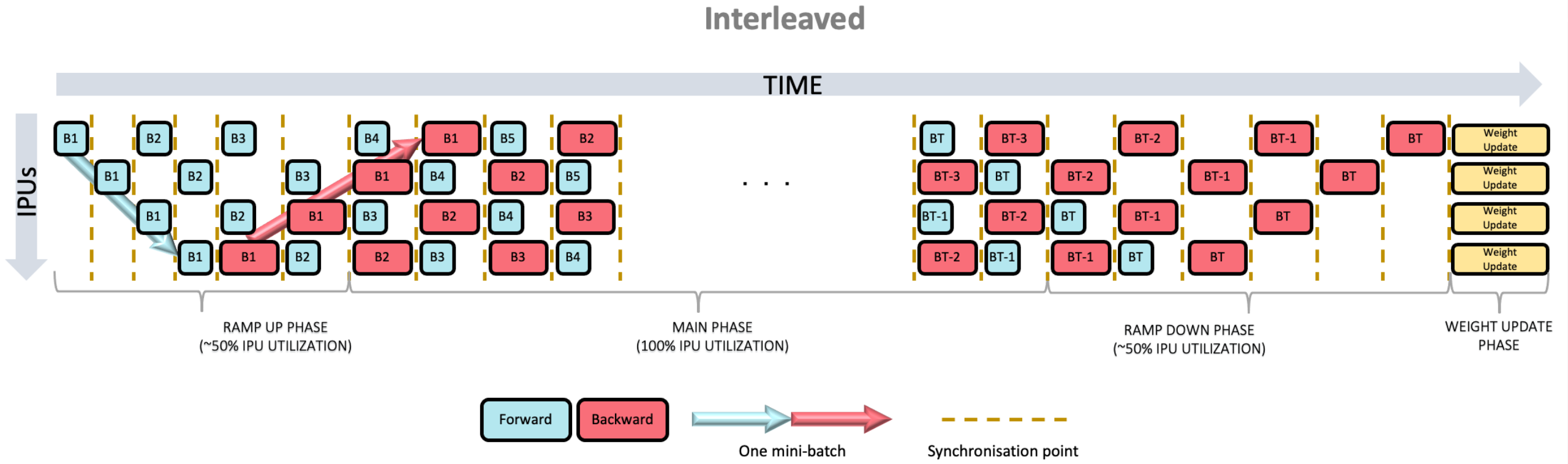




https://docs.graphcore.ai/projects/tf-model-parallelism/en/latest/pipelining.html#increase-the-gradient-accumulation-count

GRADIENT ACCUMULATION

- Pipelining 시 main execution 의 비율 확대 → 속도 향상

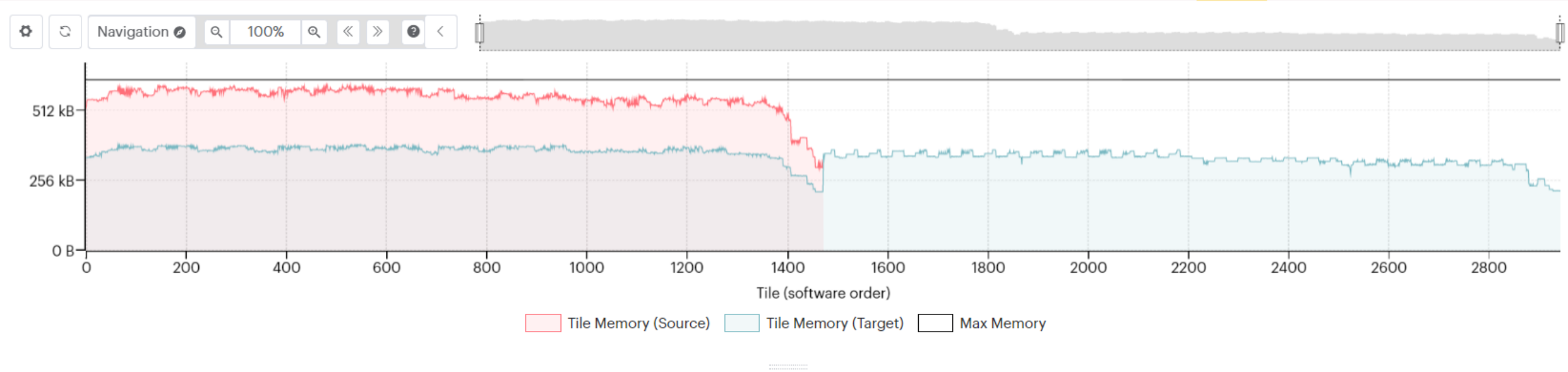


https://docs.graphcore.ai/projects/tf-model-parallelism/en/latest/pipelining.html#pipeline-scheduling

TUTORIAL

- Model Parallel

File > compare two reports



TUTORIAL

- Model Parallel + Data Parallel

```
opts = poptorch.Options()
                                    opts = poptorch.Options()
                                    opts.Training.gradientAccumulation(3)
                                    opts.replicationFactor(2)

model = torchvision.models.resnet50()
                                    model = torchvision.models.resnet50()
                                    model.layer4 = poptorch.BeginBlock(model.layer4, ipu_id=1)
```

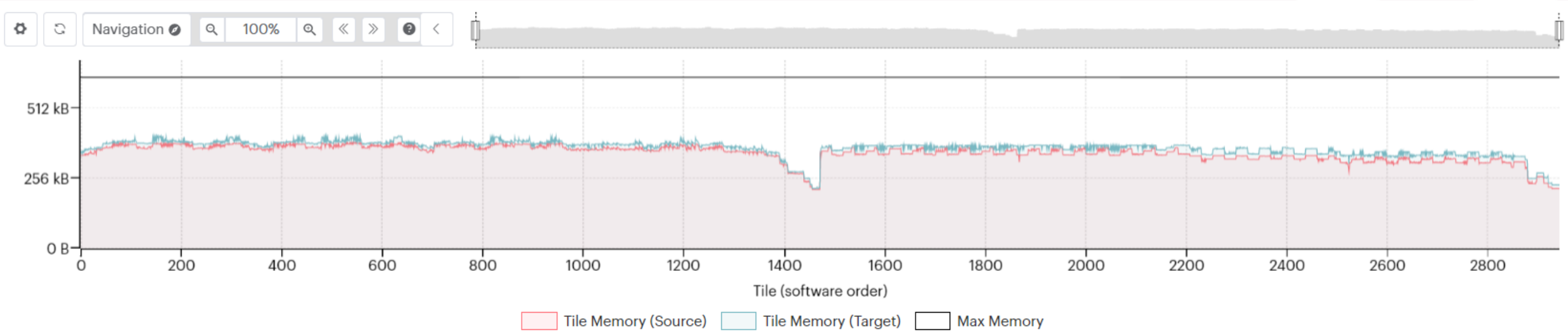
Target	
Type	IPU
Architecture	Mk2
Timestamp	2022-05-30 20:11:37
Tiles per IPU	1472
IPUs per Replica	2
Replicas	2
Total Tiles	5888
Total IPUs	4
Memory per Tile	624 kB
Memory per IPU	897 MB
Memory per Replica	1,794 MB
Total Memory	3,588 MB
Compute Set Instrumentation	Tile
External Exchange Instrumentation	Tile

Graph	
Number of compute sets	2,216
Number of edges	13,208,391
Number of variables	8,832,650
Number of vertices	4,335,377

Poplar	
Version	2.4.0 (10a96ee536)

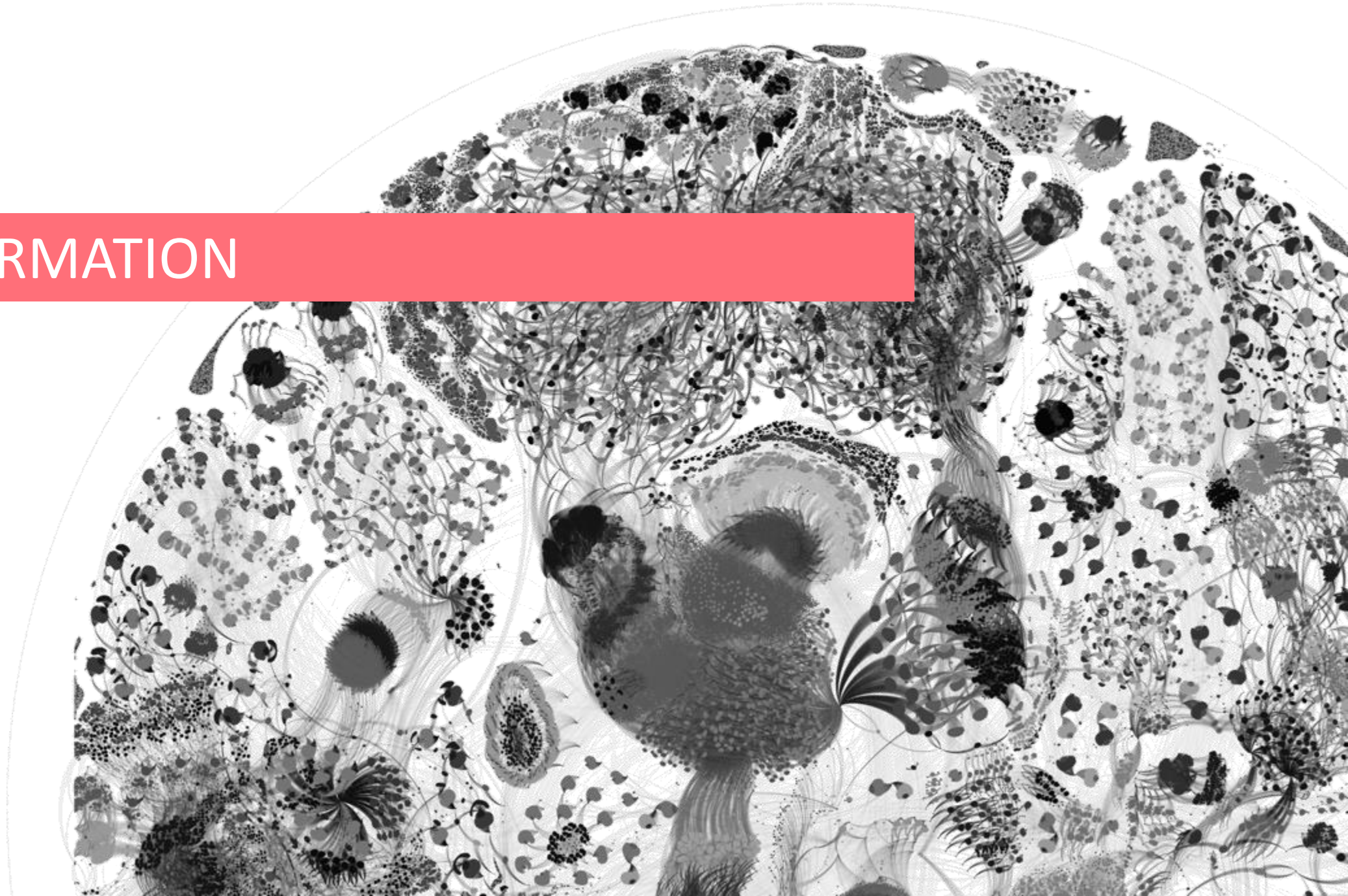
TUTORIAL

- Model Parallel + Data Parallel



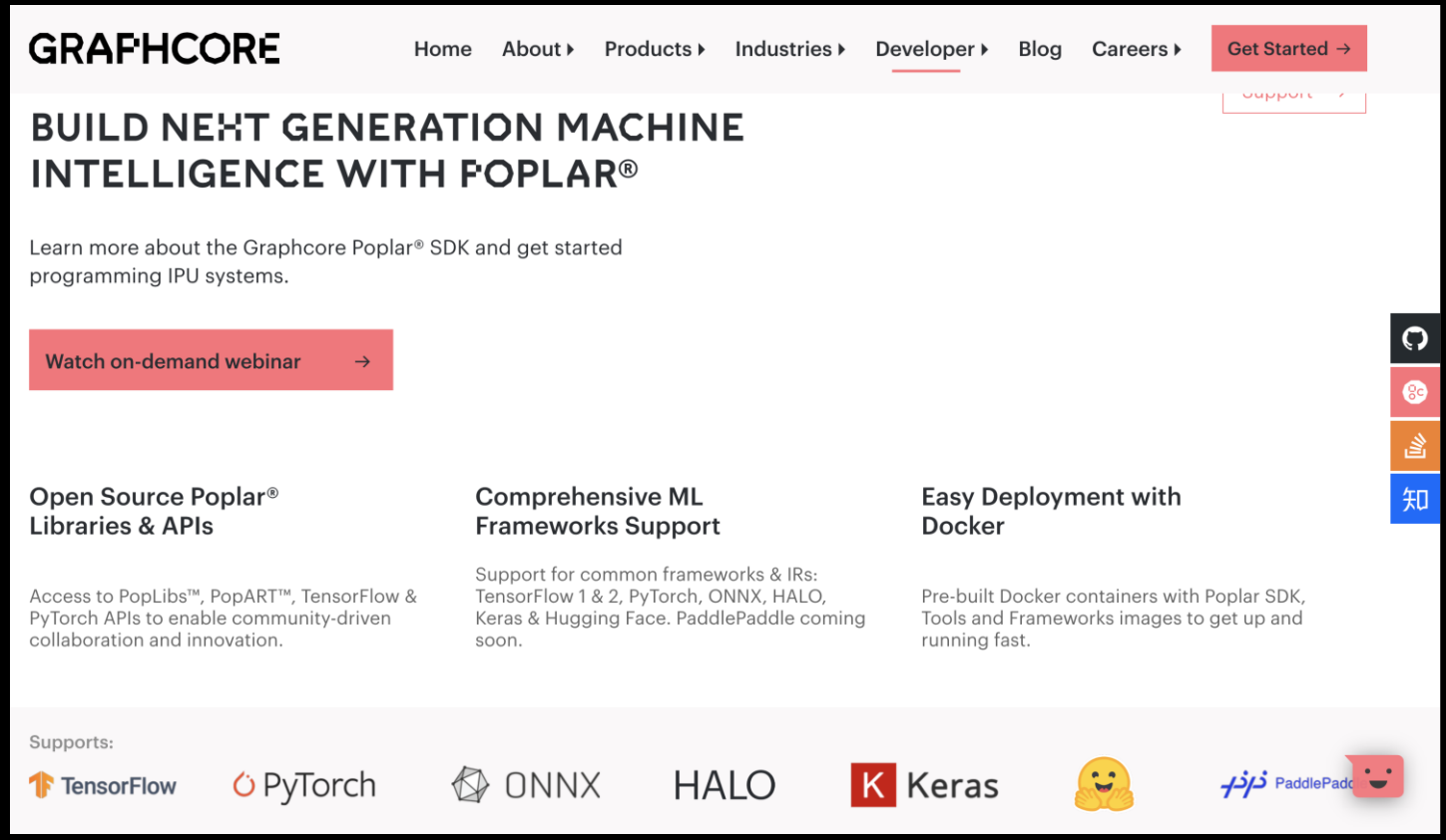
Graph	Model Parallel	Model + Data Parallel
Number of compute sets	1,994	2,216 (222)
Number of edges	12,866,597	13,208,391 (341,794)
Number of variables	7,857,014	8,832,650 (975,636)
Number of vertices	4,163,455	4,335,377 (171,922)

MORE INFORMATION



DEVELOPER PORTAL

https://graphcore.ai/developer

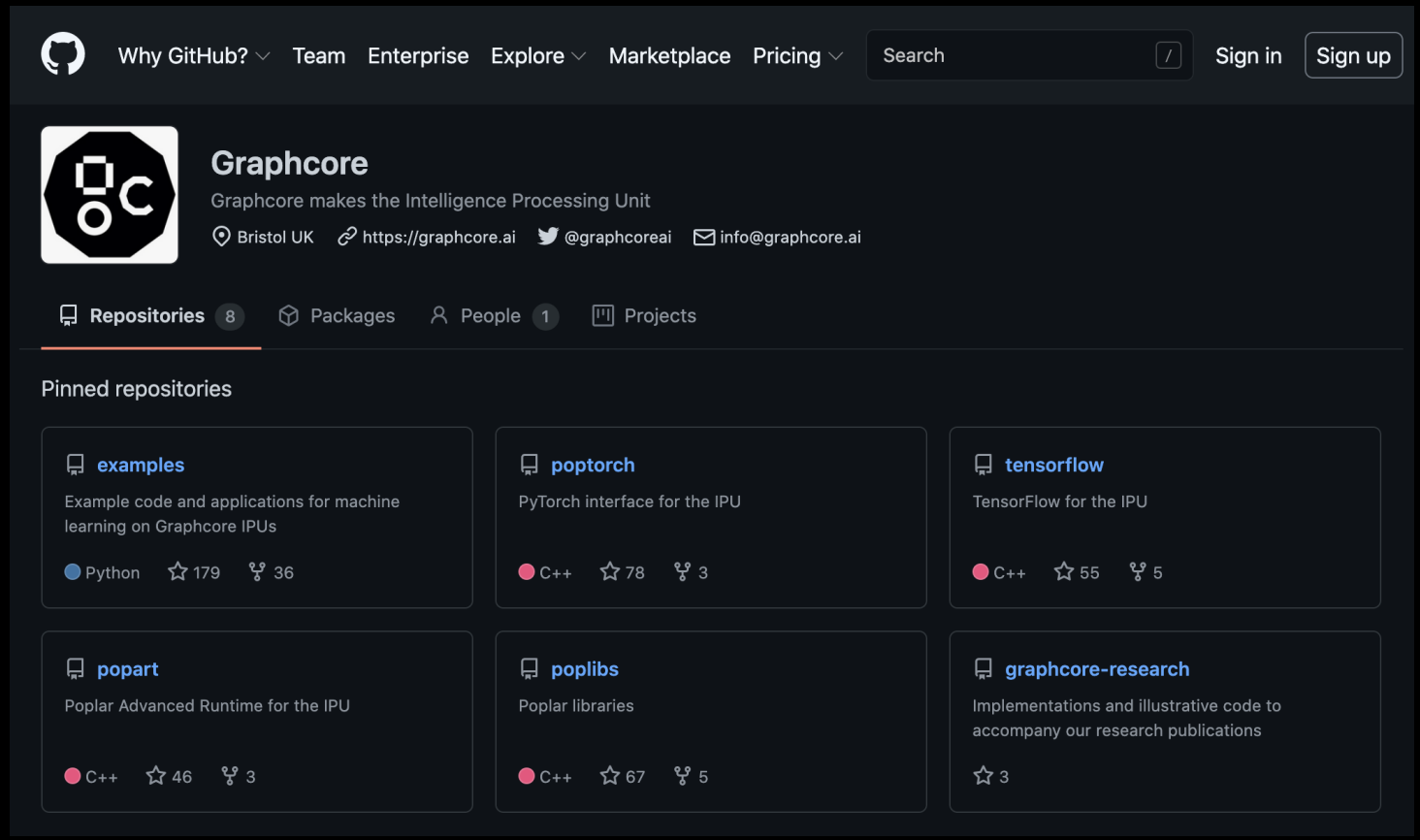


GRAPHCORE DOCUMENTS

https://docs.graphcore.ai

GRAPHCORE GITHUB

https://github.com/graphcore



THANK YOU

Minkyu Kim
minkyuk@graphcore.ai