

PROFILING WITH POPVISION

July 26, 2023

Alexander Tsyplikhin

GRAPHCORE

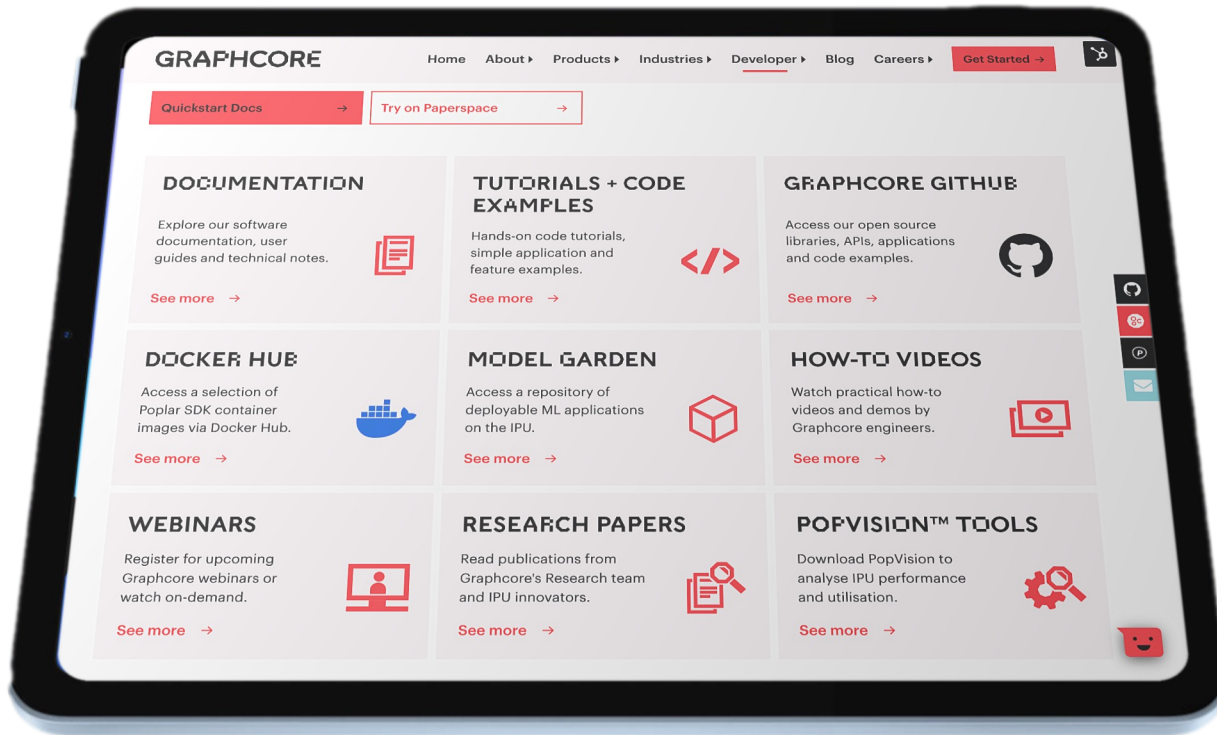


AGENDA

- PopVision Tools
 - PopVision Graph Analyser
 - PopVision System Analyser

GRAPHCORE SOFTWARE ECOSYSTEM

WORLD CLASS DEVELOPER RESOURCES FOR IPU USERS



WWW.GRAPHCORE.AI/DEVELOPER

GRAPHCORE

Graphcore Documents
Version: Latest

- Getting Started
- Software Documents
- Hardware Documents
- Technical Notes and White Papers
- Examples and Tutorials
- Document Updates
- Alphabetical List of All Documents
- Graphcore License Agreements

GRAPHCORE DOCUMENTS

Getting Started

Background information and quick-start guides for Graphcloud and Pod systems

Software Documents

Documentation for the Poplar SDK and other software

Hardware Documents

Documentation for installing and using IPU-Machines and Pod systems

Technical Notes and White Papers

Technical notes and white papers on Graphcore technology

Document Updates

The latest news about new documents and examples

Examples and Tutorials

Tutorials and application examples for running on the IPU



Getting started with PyTorch for the IPU

Running a basic model for training and inference

AI Customer Engineer, Chris Bogdiukiewicz introduces PyTorch for the IPU. With PopTorch™ - a simple Python wrapper for PyTorch programs, developers can easily run models, directly on Graphcore IPUs with a few lines of extra code.

[Get the Code](#) →

In this video, Chris provides a quick demo on running a basic model for both training and inference using a MNIST based example.

[Read the Guide](#) →





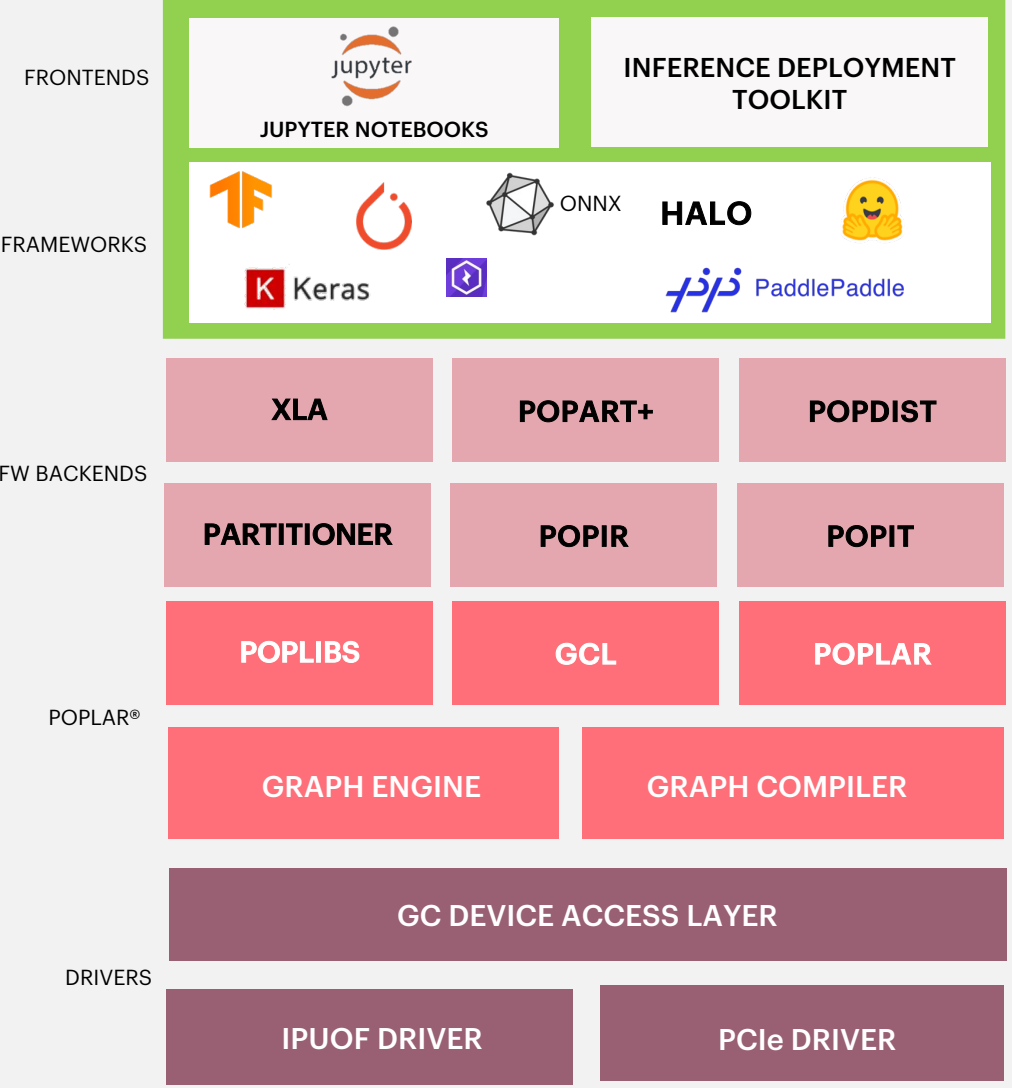
GRAPHCORE SOFTWARE

- NLP/TRANSFORMERS
- IMAGE CLASSIFICATION/CNNS
- OBJECT DETECTION
- LARGE MODELS
- MLPERF
- CONDITIONAL SPARSITY
- GNNS

ML APPLICATIONS

- TUTORIALS
- CODE EXAMPLES
- DOCUMENTATION
- VIDEOS
- NATIVE IPU CODERS PROGRAM
- APPS PORTFOLIO

DEVELOPER ECOSYSTEM



POPLAR® SDK

- GRAPH ANALYZER
- SYSTEM ANALYZER
- DEBUGGER
- DEVELOPMENT ENVIRONMENT

POPVISION TOOLS

- V-IPU
- SYSTEM MONITORING
- PROMETHEUS
- GRAFANA

JOB DEPLOYMENT

- K8S
- SLURM

SYSTEM SOFTWARE



POPVISION™ TOOLS

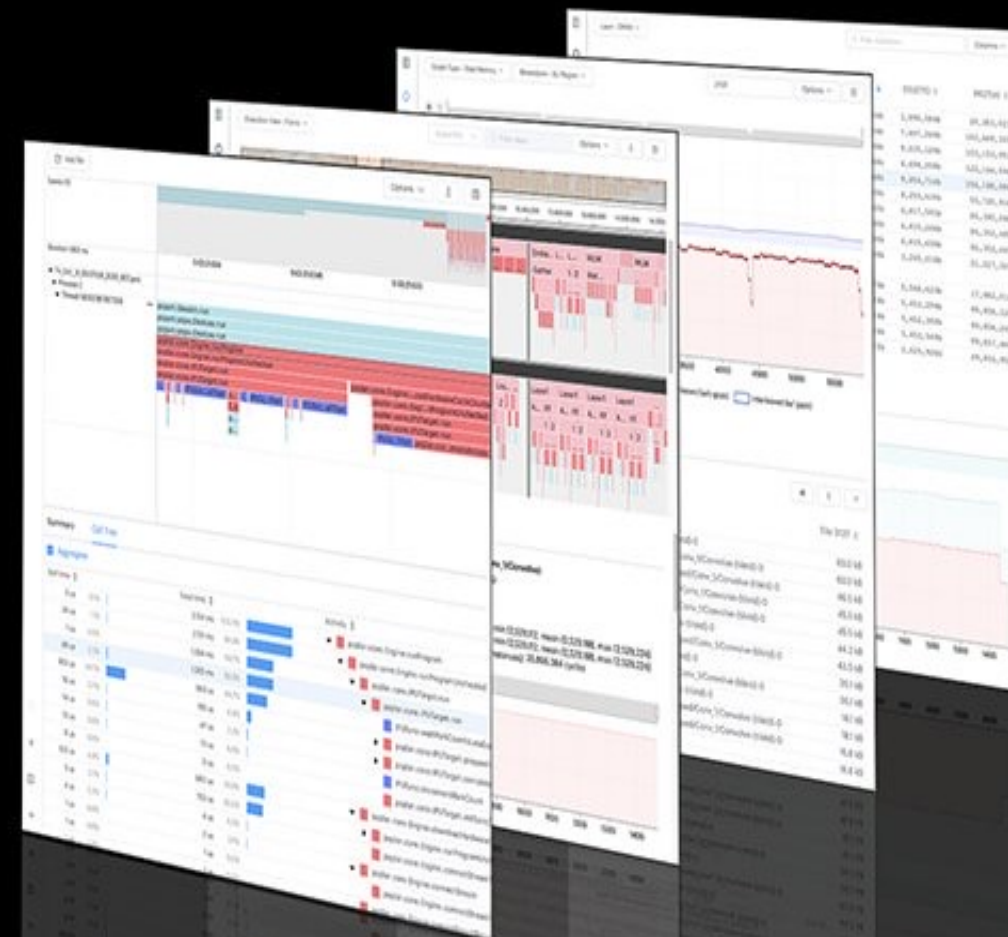
POPLAR™ POPVISION TOOLS

GRAPH ANALYSER

Useful for analysing and optimising the memory use and execution performance of ML models on the IPU

SYSTEM ANALYSER

Graphical view of the timeline of host-side application execution steps



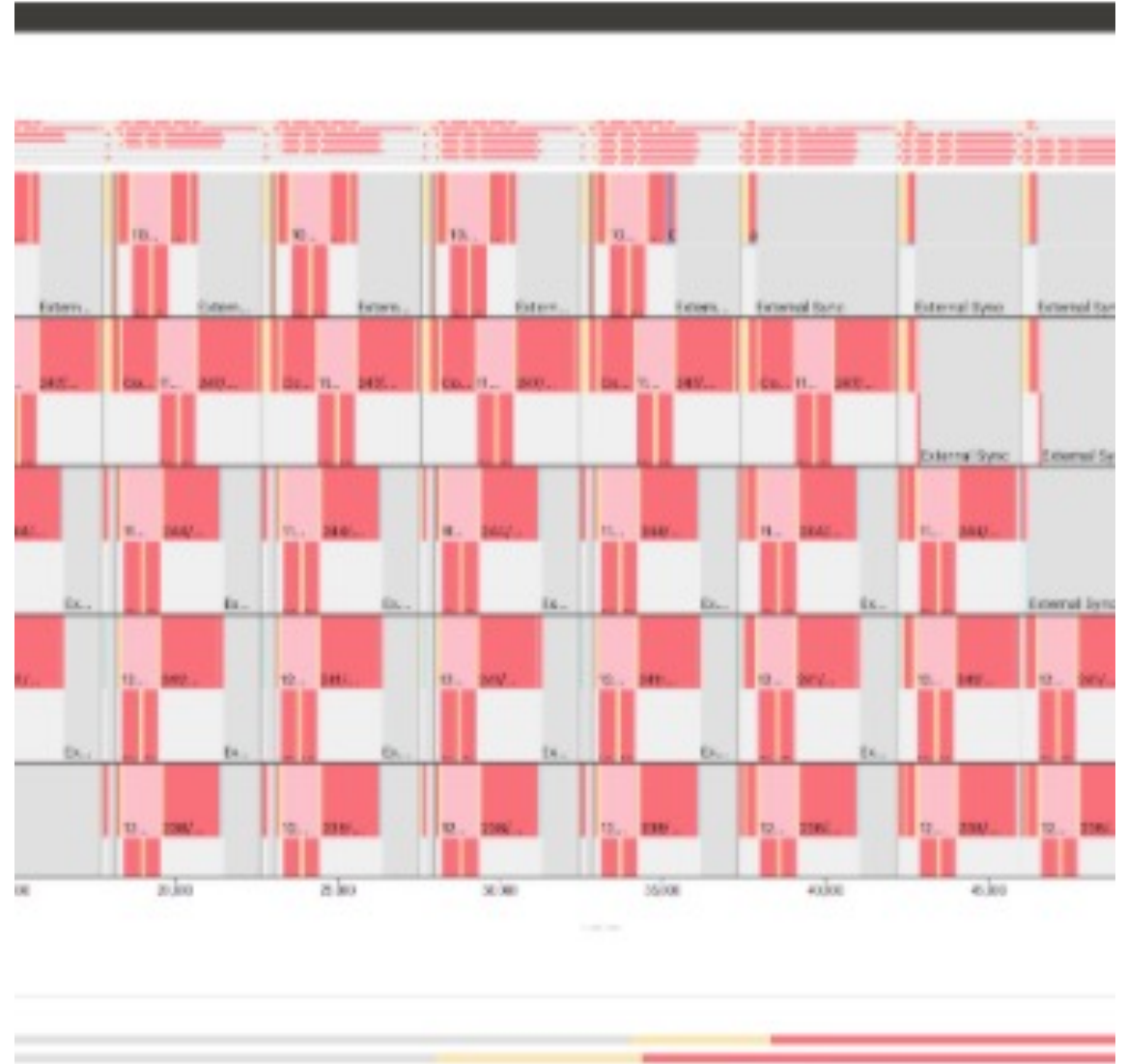
“Our team was very impressed by the care and effort Graphcore has clearly put into the PopVision graph and system analysers. It’s hard to imagine getting such a helpful and comprehensive profiling of the code elsewhere, so this was really a standout feature in our IPU experience.”



Dominique Beaini, Valence Discovery, a leader in AI-first drug design

POPVISION GRAPH ANALYSER

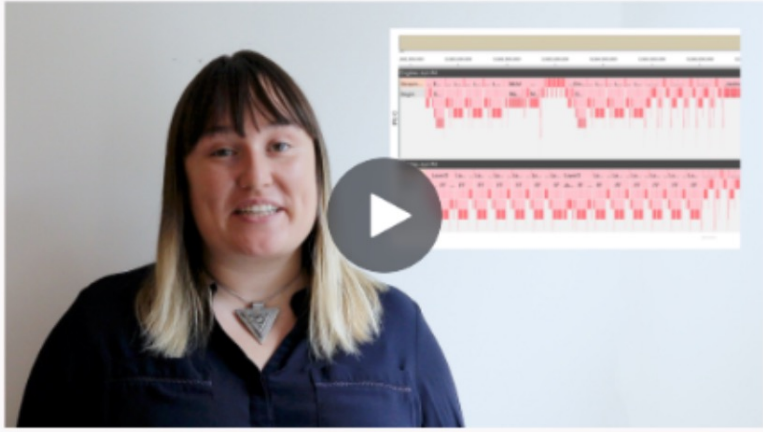
- You can use the PopVision Graph Analyser tool to debug IPU programs and generate reports on compilation and execution of the program.
- This tool can be downloaded from graphcore.ai/downloads
- There is a built-in help system within the tool for any questions you might have about producing and analysing reports.



PopVision Graph Analyser

Getting started with PopVision™

Intro to the PopVision™ Graph Analyser



Getting started video available on the developers portal



Several new features including:

- A new file format for the graph and execution profile, resulting in a 50% file size reduction
- Enhanced PopLibs debug information

Liveness Report

The debug information shown for a variable now displays enhanced information. For each variable that has debug information, you can now see the PopLibs API that created it, its arguments and its outputs.

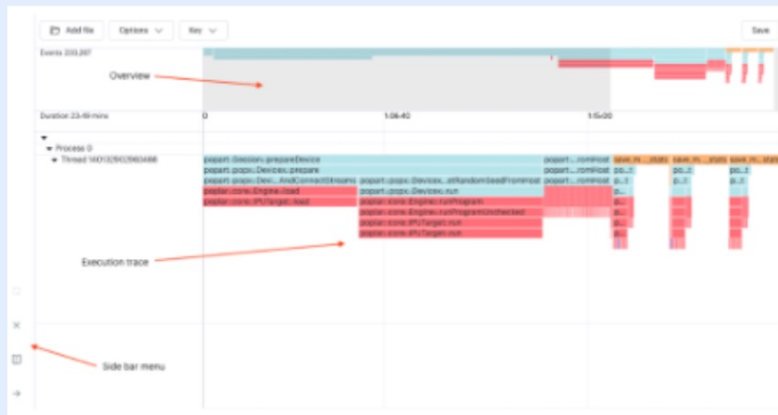
Enhanced debug information has been added to program steps. Program steps show Poplar and PopLibs debug information such as which PopLibs API created that program step, its arguments and its outputs.

Check out the integrated help or visit our developer portal for more information

PopVision System Analyser

The PopVision System Analyser allows developers to understand the execution of programs running on the host processor which control the IPU(s). The System Analyser shows the interaction between the host and the IPU(s) so that developers can understand where the bottlenecks are in the execution of their applications.

The PopVision System Analyser visualises the information collected by the PopVision Trace Instrumentation Library which is part of the Poplar SDK.



Show the execution of the software on the host processor enabling users to identify bottlenecks in execution between CPU & IPU(s).



Provide profile insights as you scale models to multiple CPUs / IPUs.

Visit our developer portal for more information and the latest documentation:

<https://www.graphcore.ai/developer>

CREATE PROFILE



```
(3.3.0+1403_poptorch) alext@bpod16:~/work/examples/tutorials/tutorials/pytorch/basics$ POPLAR_ENGINE_OPTIONS='{ "autoReport.all": "true", "autoReport.directory": "./report", "debug.allowOutOfMemory": "true", "profiler.includeFlopEstimates": "true" }' python walkthrough.py
epochs: 0% | 0/5 [00:00<?, ?it/s]
22:25:58.232 [poptorch:cpp] [warning] [DISPATCHER] Type coerced from Long to Int for tensor id 12 | 0/3750 [00:00<?, ?it/s]
Graph compilation: 100% | 100/100 [00:39<00:00]
epochs: 100% | 5/5 [01:21<00:00, 16.21s/it]
/nethome/alex/venvs/poplar_sdk-ubuntu_20_04-3.3.0+1403-208993bbb7/3.3.0+1403_poptorch/lib/python3.8/site-packages/torch/nn/modules/module.py:1802: UserWarning: Positional args are being deprecated, use kwargs instead. Refer to https://pytorch.org/docs/master/generated/torch.nn.Module.html#torch.nn.Module.state_dict for details.
  warnings.warn(
Graph compilation: 100% | 100/100 [00:19<00:00]
Eval accuracy: 89.25%
Graph compilation: 100% | 100/100 [00:09<00:00]
tensor([[ -4.4754, -0.9084, -3.2791, -5.8905, -3.4738, -1.4554, -1.8643, -6.6169,
          -2.1170, -5.5300]])
IPU predicted class: Trouser
CPU predicted class: Trouser
```



Summary

Insights

Memory Report

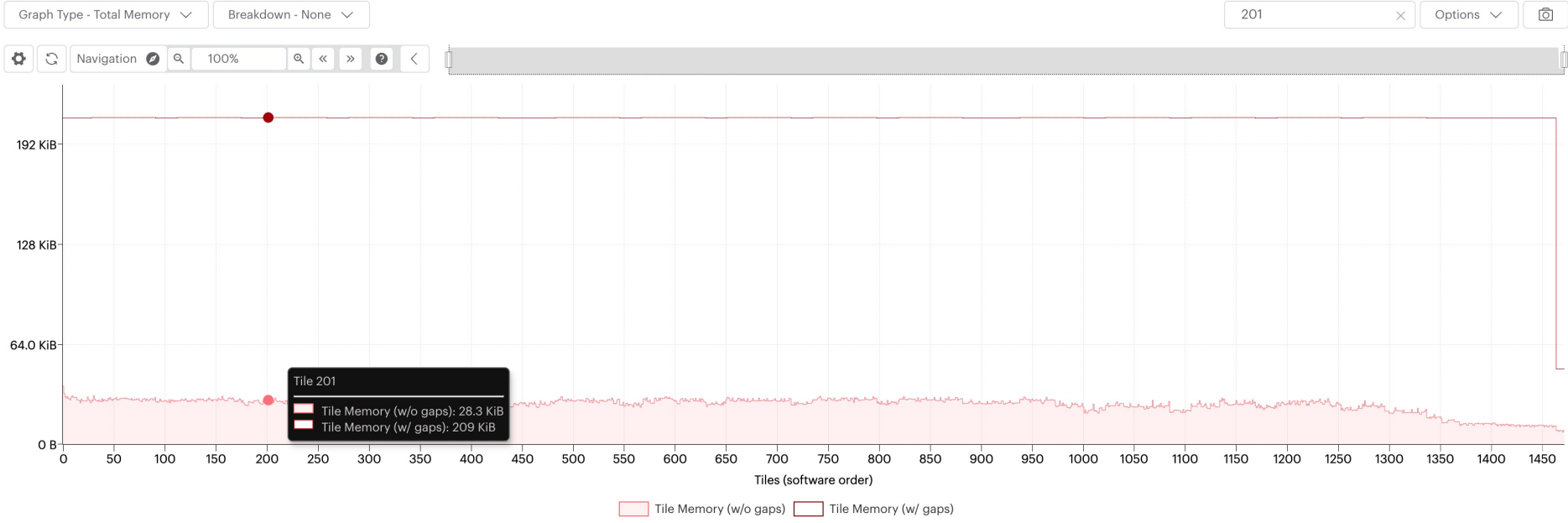
Liveness Report

Program Tree

Operations Summary

Operations Graph

Execution Trace



Details | Compute Sets | Vertices | Exchanges | Variables

	Tile 201
Physical Tile Id	522
Memory Including Gaps	209 KiB
Memory Excluding Gaps	28.3 KiB
By Memory Region	
Non-interleaved	27.4 KiB
Interleaved	960 B
Overflowed	0 B
By Data Type	
Not Overlapped	
Variables	1.6 KiB
Constants	414 B
Control Code	2.4 KiB
Vertex Code	8.8 KiB
Internal Exchange Code	7.2 KiB
Host Exchange Code	640 B
Global Exchange Code	0 B
Vertex Instances	1.4 KiB

Reload Report

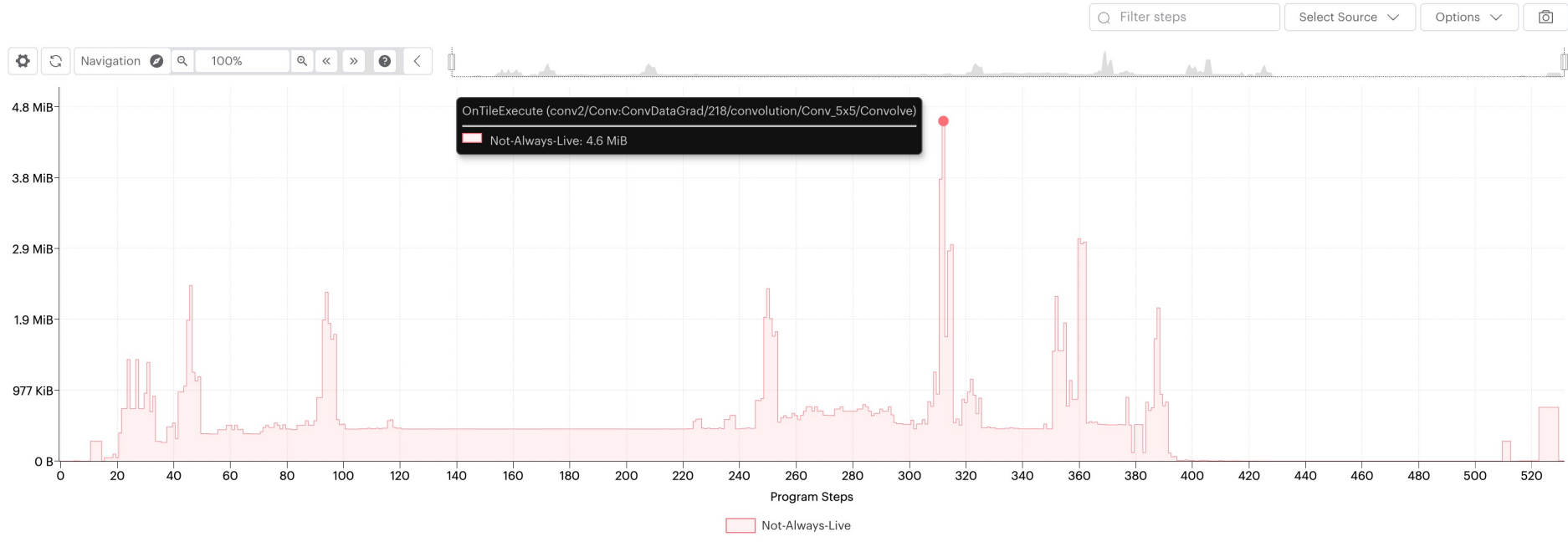
Close Report

Documentation

Minimise



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



Selected Program Steps

Step 312 OnTileExecute (conv2/Conv:ConvDataGrad/218/convolution/Conv_5x5/Convolve)

- Always-Live Variables
- Not-Always-Live Variables
- Vertices
- Cycle Estimates
- FLOP Estimates

Filter variables

Variable	All Tiles
Total	29.3 MiB
vertexCode	11.8 MiB
internalExchangeCode	8.5 MiB
controlCode	3.0 MiB
vertexInstanceState	1.9 MiB
hostExchangeCode	924 KiB
stack	535 KiB
instrumentationResults	402 KiB
▶ Accl1__fc1.weight	380 KiB
▶ fc1.weight	380 KiB
vertexFieldData	241 KiB
copyDescriptor	223 KiB
hostExchangePacketHeader	198 KiB

- Reload Report
- Close Report
- Documentation
- Minimise



Summary

Insights

Memory Report

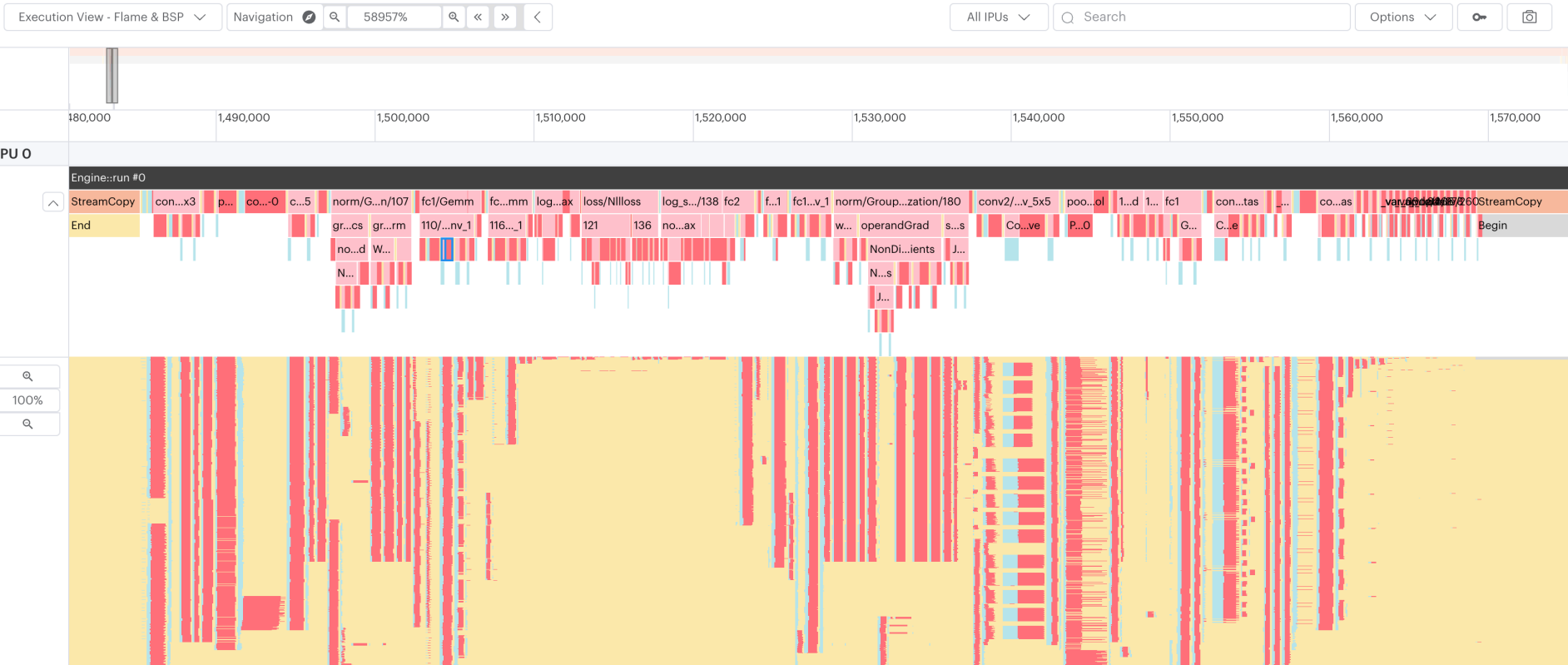
Liveness Report

Program Tree

Operations Summary

Operations Graph

Execution Trace



Summary

Details

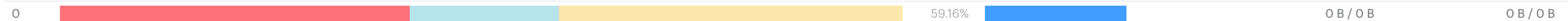
Total Cycles 94,526 Total FLOPs 31,844,509

IPU Proportions

Tile Utilisation 7

Stream Copy (Rx/Tx)

Global Exchange (Rx/Tx)



! One or more step types are hidden. These step types are excluded from the tile balance, tile utilisation, and cycle proportions shown and so the statistics may not be representative. Step types can be shown or hidden using the view options dropdown.

Reload Report

Close Report

Documentation

Minimise



POPVISION TUTORIALS

Continued in the repositories below (follow the READMEs)

[tutorials/tensorflow2/infeed_outfeed](#)

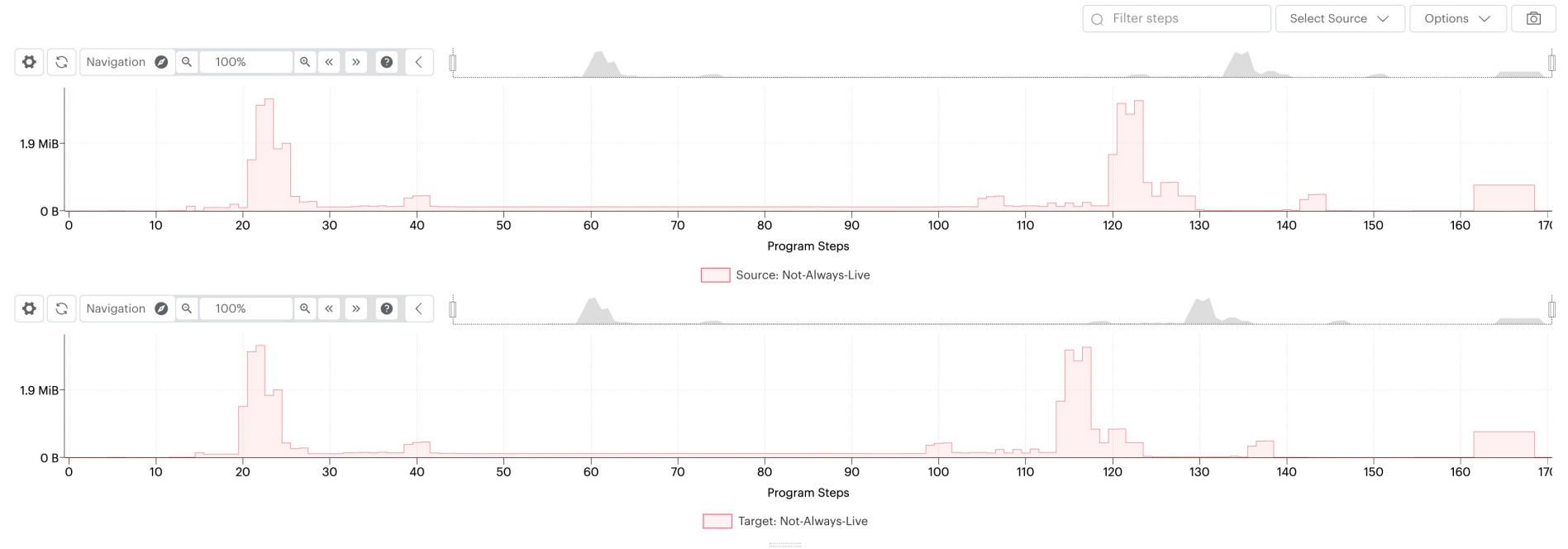
[tutorials/pytorch/pipelining](#)

[tutorials/popvision/system_analyser_instrumentation](#)



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

- Swap Reports
- Reload Report
- Close Report
- Documentation
- Minimise



Selected Program Step
 Name (source): None selected. Click on the graph to select a source program step
 Name (target): None selected. Click on the graph to select a target program step

[Always-Live Variables](#) [Not-Always-Live Variables](#) [Vertices](#) [Cycle Estimates](#)

Filter variables

Variable	Source: All Tiles	Target: All Tiles	Diff: All Tiles
Total	11.3 MiB	11.2 MiB	-113 KiB
vertexCode	6.0 MiB	6.0 MiB	-16 B (-0.00%)
controlCode	1.5 MiB	1.5 MiB	-37.7 KiB (-2.41%)
internalExchangeCode	1.2 MiB	1.1 MiB	-47.6 KiB (-3.99%)
hostExchangeCode	595 KiB	556 KiB	-39.2 KiB (-6.58%)
stack	540 KiB	529 KiB	-11.0 KiB (-2.04%)
vertexInstanceState	527 KiB	527 KiB	20 B (0.00%)
while/sequential/dense/MatMul/dot.35/rhs	392 KiB	—	-392 KiB (-100.00%)
sequential/dense/MatMul/dot/rhs	—	392 KiB	392 KiB (100.00%)
instrumentationResults	180 KiB	180 KiB	4 B (0.00%)
hostExchangePacketHeader	114 KiB	137 KiB	23.7 KiB (17.25%)



Summary

Insights

Memory Report

Liveness Report

Program Tree

Operations Summary

Operations Graph

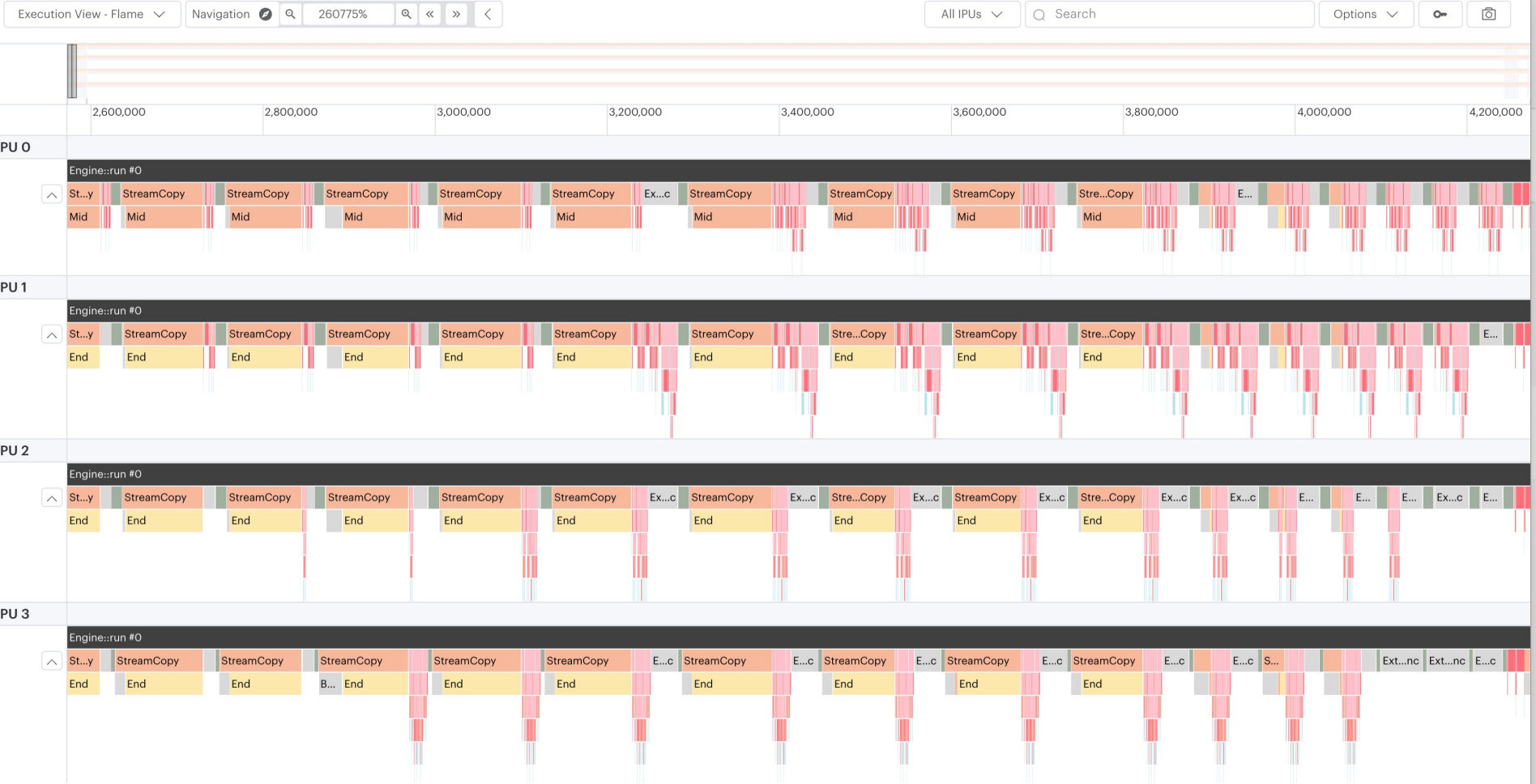
Execution Trace

Reload Report

Close Report

Documentation

Minimise

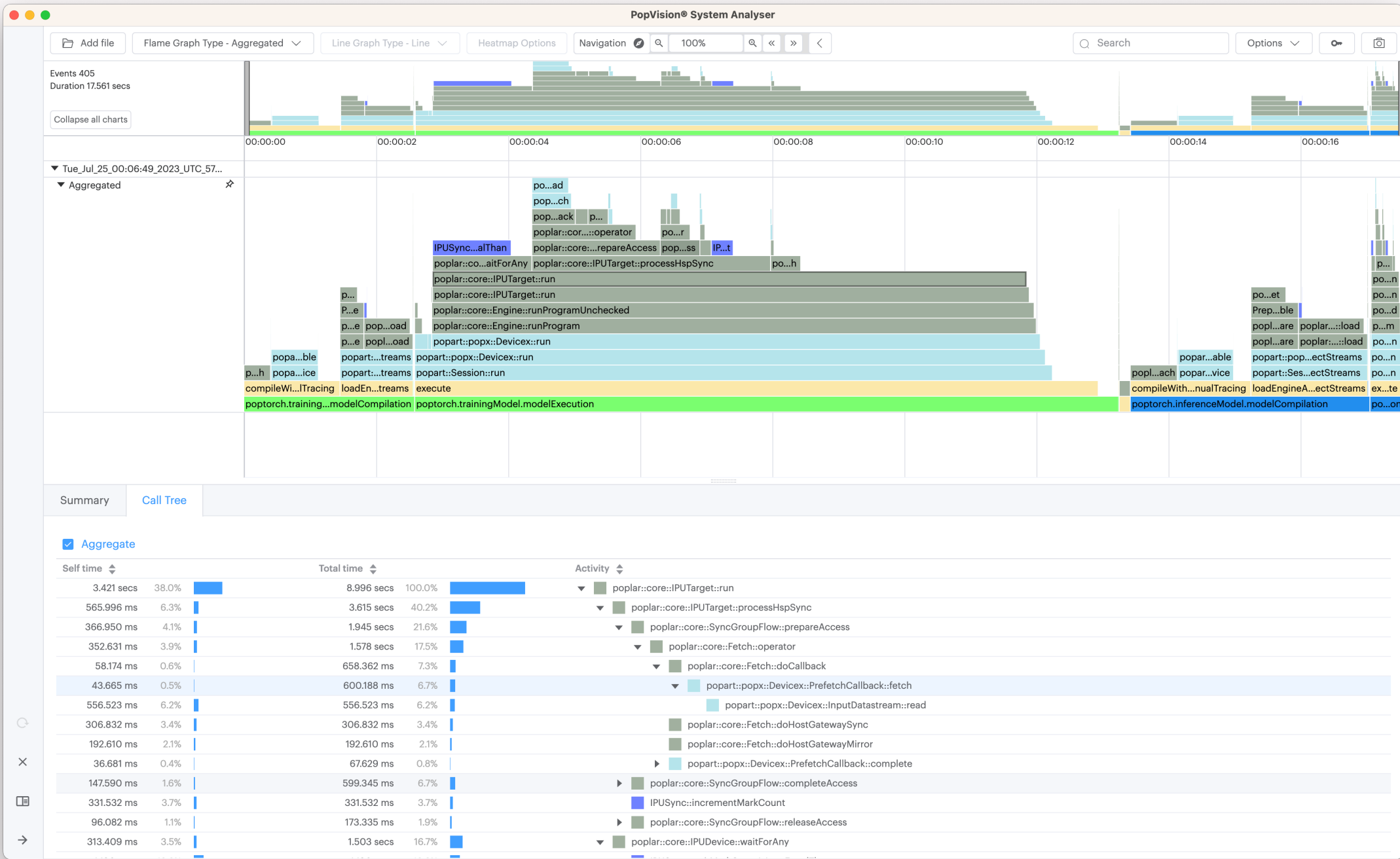


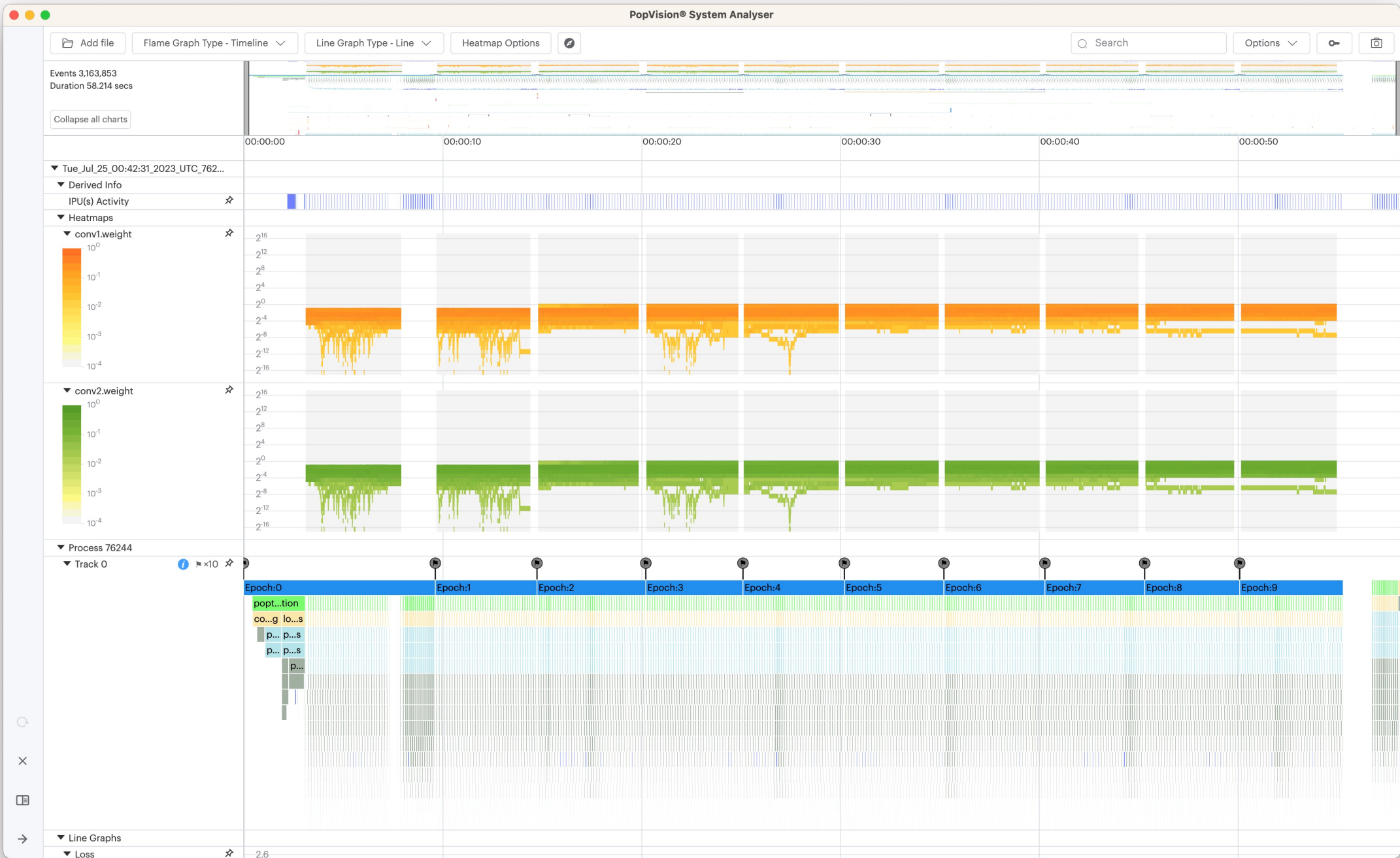
Summary Details

Total Cycles 1,700,636 Total FLOPs 2,227,345,013

IPU	Proportions	Tile Utilisation	Stream Copy (Rx/Tx)	Global Exchange (Rx/Tx)
0		56.02%	1.1 MiB / 172 KiB	4.1 MiB / 4.2 MiB
1		54.73%	0 B / 0 B	5.0 MiB / 5.3 MiB
2		26.01%	0 B / 0 B	1.8 MiB / 1.4 MiB
3		18.01%	768 B / 320 B	640 KiB / 640 KiB





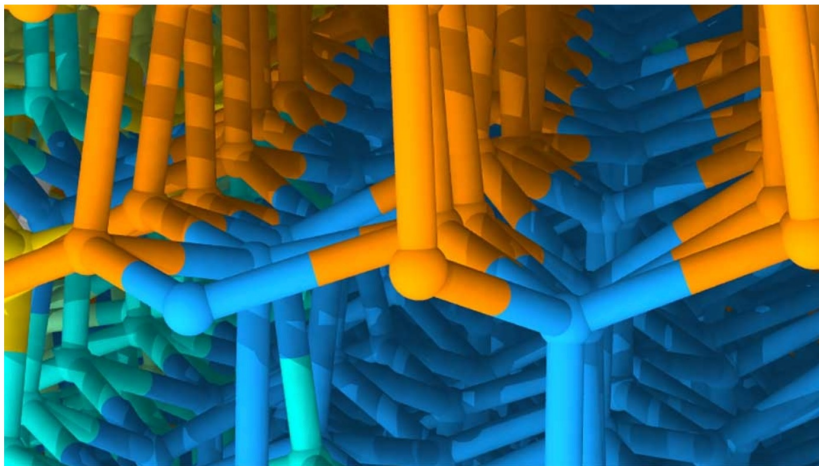


APPLY AND JOIN TODAY



Director's Discretionary Allocation Program

The ALCF Director's Discretionary program provides "start up" awards to researchers working to achieve computational readiness for for a major allocation award.



Molecular dynamics simulations based on machine learning help scientists learn about the movement of the boundary between ice grains (yellow/green/cyan) and the stacking disorder that occurs when hexagonal (orange) and cubic (blue) pieces of ice freeze together. Image: Henry Chan and Subramanian Sankaranarayanan, Argonne National Laboratory

Apply at alcf.anl.gov/science/directors-discretionary-allocation-program

general



charlieb 6:05 AM

Pleased to share with you all some new work from the Graphcore research team! 🎉

Our paper *Unit Scaling* introduces a new method for low-precision number formats, making FP16. We've managed to train BERT in these formats for the first time without loss scaling.

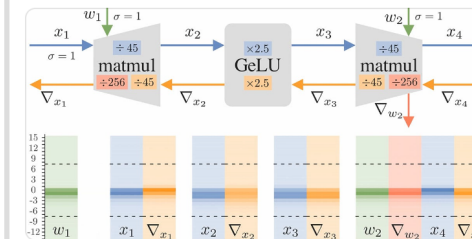
- You can find our blog post here: <https://www.graphcore.ai/posts/simple-fp16-and-fp8-training>
- Paperspace notebook (try it yourself!): <https://ipu.dev/qXfm2a>
- Arxiv paper: <https://arxiv.org/abs/2303.11257>

(& we were also featured on Davis Blalock's popular [ML newsletter](#) this week) (edited)

graphcore.ai

Simple FP16 and FP8 training with unit scaling

Unit Scaling is a new low-precision machine learning method able to train language models in FP16 and FP8 without loss scaling. (69 kB)



arXiv.org

Unit Scaling: Out-of-the-Box Low-Precision Training

We present unit scaling, a paradigm for designing deep learning models that simplifies the use of low-precision number formats. Training in FP16 or the recently proposed FP8 formats offers substantial efficiency gains, but can lack sufficient range for out-of-the-box training. Unit scaling addresses this by introducing a principled approach to model numerics: seeking unit variance of

Show more



Join at graphcore.ai/join-community

TUESDAY, 25 JULY



- 1:30 PM** → 1:45 PM **Introduction** ⌚ 15m
- 1:45 PM** → 2:15 PM **Graphcore BowPod64 Hardware** ⌚ 30m
- 2:15 PM** → 3:00 PM **Software Stack: TensorFlow, PyTorch, and Poplar** ⌚ 45m
- 3:00 PM** → 3:15 PM **Break** ⌚ 15m
- 3:15 PM** → 3:45 PM **Porting applications with Poplar** ⌚ 30m
- 3:45 PM** → 4:30 PM **How to use Bow Pod64@ ALCF** ⌚ 45m

WEDNESDAY, 26 JULY



- 1:30 PM** → 2:15 PM **Deep Dive on Graph neural networks and Large Language Models** ⌚ 45m
- 2:15 PM** → 2:45 PM **Profiling with PopVision** ⌚ 30m
- 2:45 PM** → 3:00 PM **Break** ⌚ 15m
- 3:00 PM** → 3:45 PM **Hands-on session** ⌚ 45m
- 3:45 PM** → 4:15 PM **Best Practices, Q&A** ⌚ 30m



THANK YOU

Alexander Tsyplikhin
alex@graphcore.ai