R244 · SESSION 6 · OPTIMISATION IN COMPUTER SYSTEMS

# Unity: Joint Optimization of Algebraic Transformations and Parallelization

C. Unger, Z. Jia, W. Wu, S. Lin, M. Baines, … A. Aiken

Stanford · CMU · Meta · LANL and others · OSDI 2022

**Joshua Fehn · 18 November 2026**

TODAY'S SESSION

# From where to run it, to what to run

PAPERS 1–2 · PLACEMENT WITH RL, PLACETO

**Where does each operator run?**

The graph is fixed. A learned policy places it on devices.

PAPER 3 · UNITY

**Which equivalent graph, and how is it parallelised?**

Rewrites, parallelisation and placement are chosen together.

PAPER 4 · KERNELCRAFT

**Can LLM agents write the kernel code?**

A benchmark: agents write assembly kernels for new accelerators.

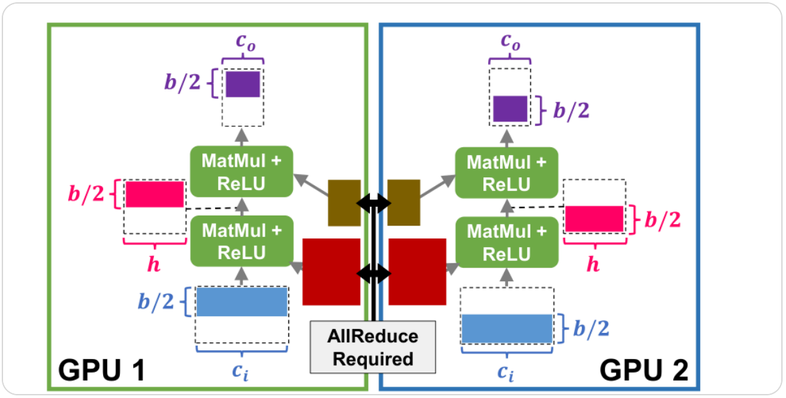
**Unity is the non-learning paper of the session:** verified graph rewrites and cost-guided search, with no RL policy and no learned cost model.

→ NEXT: WHY CHOOSING IN TWO STEPS IS NOT ENOUGH

WHAT IS OPTIMISED?

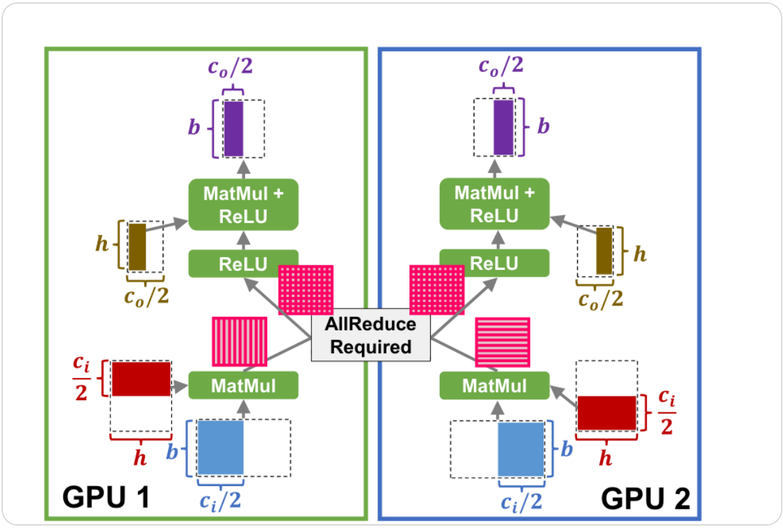
# Optimising in two steps misses the best plan

### SEQUENTIAL: FUSE FIRST, THEN PARALLELISE




Data parallelism everywhere: both weight gradients are synchronised.

### JOINT: LEAVE THE FIRST PAIR UNFUSED




Reduction parallelism on the first MatMul: only its output is synchronised.

**6×**

less communication between the GPUs in this 2-layer MLP

## Algebraic transformation

Replace part of the graph with an equivalent one, e.g. fuse MatMul + ReLU.

## Parallelisation

Split work across devices: data, model, spatial, reduction, pipeline, operator-specific.

→ NEXT: HOW CAN ONE SEARCH CHOOSE BOTH? ONE GRAPH.

Figure 2 of the paper. MNIST-sized MLP on 2 GPUs.

HOW IS THE SEARCH SPACE REPRESENTED?

# One graph for computation and parallelism

**Parallel computation graph (PCG)** EXAMPLE: A DATA-PARALLEL MATMUL




## Parallelisation operators

Six new graph nodes, in three pairs:

| | |
|---|---|
| **Partition / Combine** | split, merge |
| **Replicate / Reduce** | copy, sum |
| **Pipeline / Batch** | micro-batches |

Communication becomes visible in the graph.

## Everything is a substitution

**Substitution** = rewrite rule: swap a subgraph for an equivalent one.

Fusing and parallelising are both substitutions.

TASO's idea (Session 2: TENSAT, X-RLflow), extended to parallelism.

## Placement is kept separate

The PCG says **how** each operator is split.

It does not say **which device** runs it.

That machine mapping is found during the search.

→ NEXT: WHAT DOES A SUBSTITUTION LOOK LIKE?

HOW IS THE SEARCH SPACE REPRESENTED? · THE MOVES

# A substitution swaps a subgraph for an equivalent one

### ALGEBRAIC · AS IN TASO

MatMul → ReLU

is equivalent to

MatMul + ReLU

Two operators become one fused operator. Cheaper on a single device.

### PARALLEL · NEW IN UNITY

ReLU

is equivalent to

Partition → ReLU → Combine

States that ReLU may be split along a dimension. Most new substitutions are of this kind.

### HYBRID · NEW IN UNITY

Add

is equivalent to

Concat → Reduce

Two tensors on different devices are summed without first sending both to one device.

The search applies one substitution at a time. There are 743 from TASO and 651 new ones in Unity.

→ NEXT: NOBODY WROTE THESE BY HAND. WHERE DO THEY COME FROM?

HOW IS OPTIMISATION PERFORMED? · OFFLINE, ONCE

# Generated and proven, not hand-written

| Step | TASO (2019) | Unity adds |
|---|---|---|
| 1 · Enumerate small graphs | all graphs of up to 4 operators | the six parallelisation operators |
| 2 · Find candidate pairs | same hash of the outputs on fixed test inputs | the hash also records how each tensor dimension is split |
| 3 · Prove each pair with Z3 | from 43 hand-written operator properties | 33 more properties, for parallelism |
| Result | 743 substitutions | 651 new substitutions, in 30 min, offline |

People write the basic facts about operators. The machine finds the substitutions and proves them.

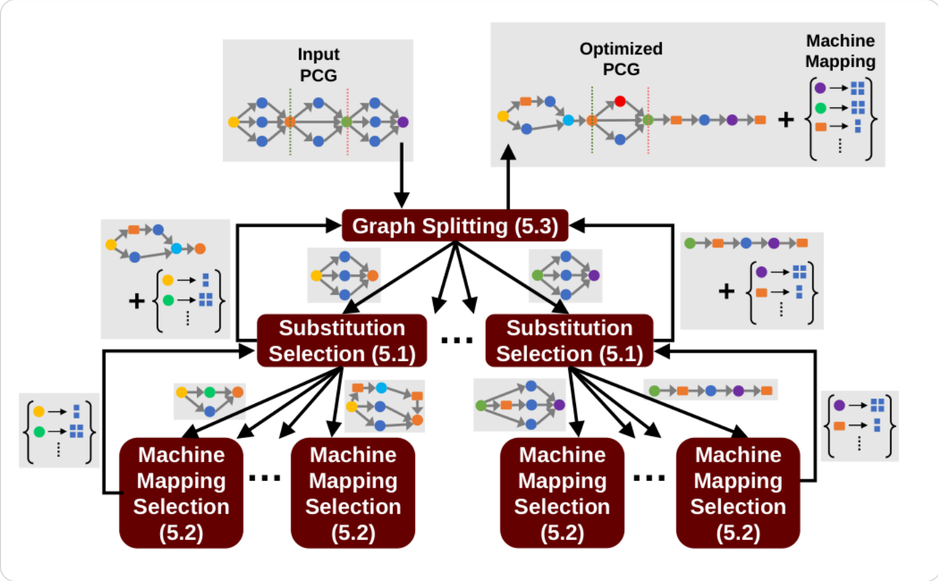
Why it matters: a new operator or form of parallelism needs a specification, not hand-written optimisation rules.

→ NEXT: WHICH SUBSTITUTIONS TO APPLY, AND WHERE? THE SEARCH.

TASO: Jia et al., SOSP 2019. Unity: Section 4 of the paper. A matching hash is only a filter; the guarantee comes from the proof.

HOW IS OPTIMISATION PERFORMED? · PER MODEL

# A three-level search keeps the joint space tractable




## 1 · Graph splitting

Cut the model into sequential parts and optimise each part separately.

## 2 · Substitution selection

TASO’s cost-based backtracking search: apply substitutions, keep candidates within 1.05× of the best.

## 3 · Machine mapping = device placement

Solved by dynamic programming for every candidate, with a cache for unchanged subgraphs. No learned policy as in papers 1–2.

Cost of a candidate: estimated compute and communication time, with methods similar to FlexFlow. No learned model.

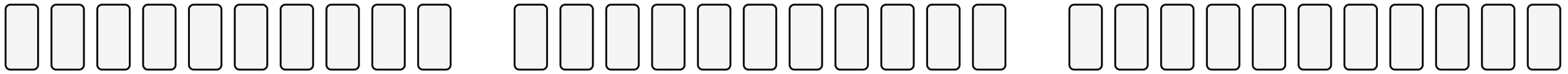
→ NEXT: A CLOSER LOOK AT THE THREE LEVELS, THEN THE RESULTS

Figure 12 of the paper. Search time for ResNeXt-50 on 6 GPUs: 57 s with splitting, cache and dynamic programming; 4 min without splitting; 37 min without splitting and cache; no result without dynamic programming (Table 2).

HOW IS OPTIMISATION PERFORMED? · LEVEL 1 WRAPS LEVEL 2

# Level 1: cut the model, then search each part

Input graph



> **The model**
>
> 32 operators in a chain. Too many for one search: candidates grow exponentially.

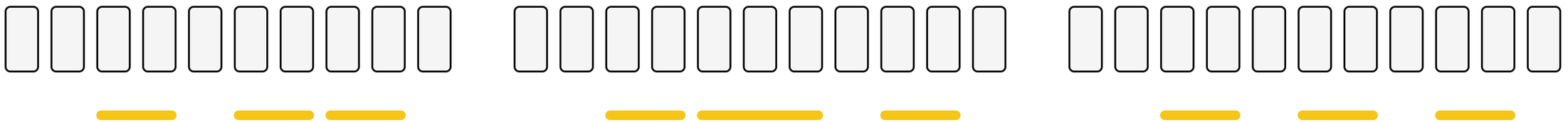
→ NEXT: INSIDE ONE PART, LEVELS 2 AND 3

Illustration with made-up operators, matches, cuts and shapes. The minimum part size k = 10 and the cut criterion are from Section 5.3.

HOW IS OPTIMISATION PERFORMED? · LEVEL 1 WRAPS LEVEL 2

# Level 1: cut the model, then search each part

Input graph



**The model**

32 operators in a chain. Too many for one search: candidates grow exponentially.

**1 · Match**

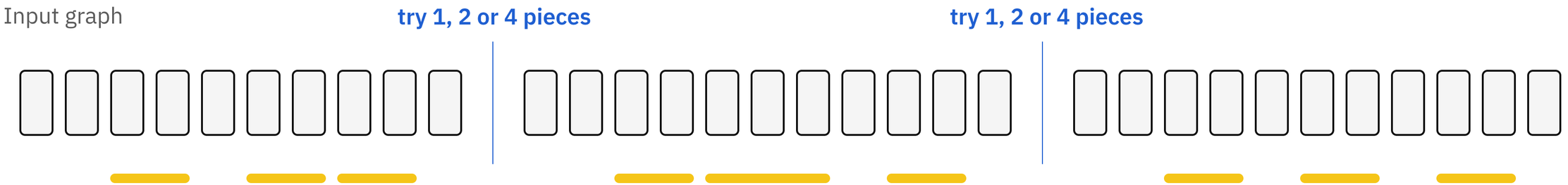
Find where rule patterns occur in the whole graph (yellow).

→ NEXT: INSIDE ONE PART, LEVELS 2 AND 3

Illustration with made-up operators, matches, cuts and shapes. The minimum part size k = 10 and the cut criterion are from Section 5.3.

HOW IS OPTIMISATION PERFORMED? · LEVEL 1 WRAPS LEVEL 2

# Level 1: cut the model, then search each part




### The model

32 operators in a chain. Too many for one search: candidates grow exponentially.

### 1 · Match

Find where rule patterns occur in the whole graph (yellow).

### 2 · Cut

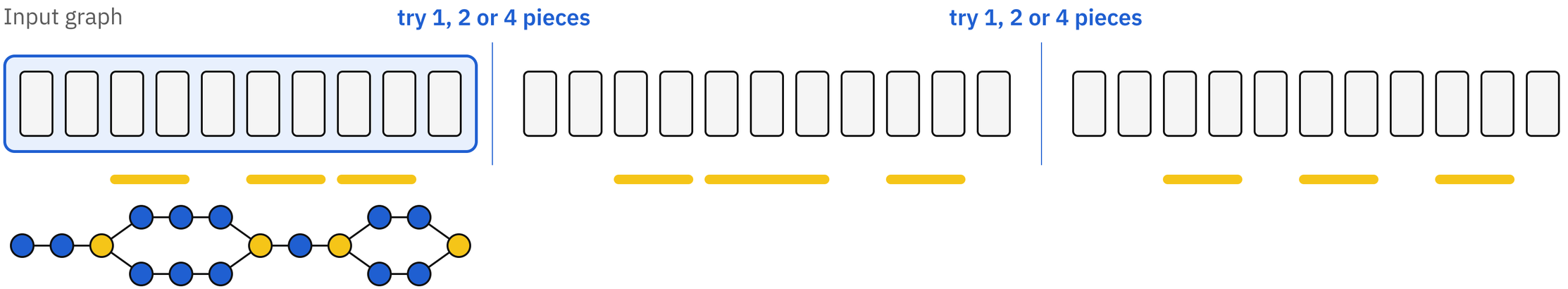
Parts of at least 10 operators, cutting the fewest matches. At each cut, try each way to partition the tensor.

→ NEXT: INSIDE ONE PART, LEVELS 2 AND 3

Illustration with made-up operators, matches, cuts and shapes. The minimum part size k = 10 and the cut criterion are from Section 5.3.

HOW IS OPTIMISATION PERFORMED? · LEVEL 1 WRAPS LEVEL 2

# Level 1: cut the model, then search each part




### The model

32 operators in a chain. Too many for one search: candidates grow exponentially.

### 1 · Match

Find where rule patterns occur in the whole graph (yellow).

### 2 · Cut

Parts of at least 10 operators, cutting the fewest matches. At each cut, try each way to partition the tensor.

### 3 · Search

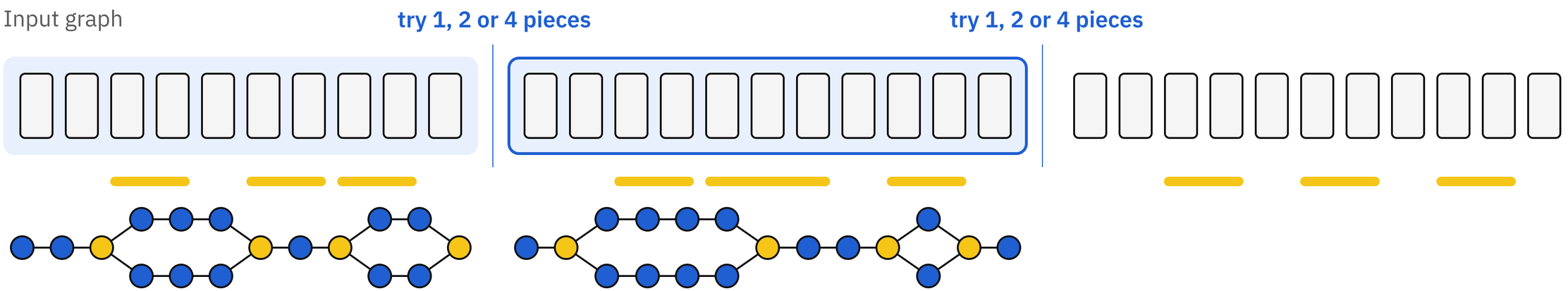
Level 2 runs on each part separately. The rewritten parts, joined, are the optimised graph.

→ NEXT: INSIDE ONE PART, LEVELS 2 AND 3

Illustration with made-up operators, matches, cuts and shapes. The minimum part size k = 10 and the cut criterion are from Section 5.3.

HOW IS OPTIMISATION PERFORMED? · LEVEL 1 WRAPS LEVEL 2

# Level 1: cut the model, then search each part




### The model

32 operators in a chain. Too many for one search: candidates grow exponentially.

### 1 · Match

Find where rule patterns occur in the whole graph (yellow).

### 2 · Cut

Parts of at least 10 operators, cutting the fewest matches. At each cut, try each way to partition the tensor.

### 3 · Search

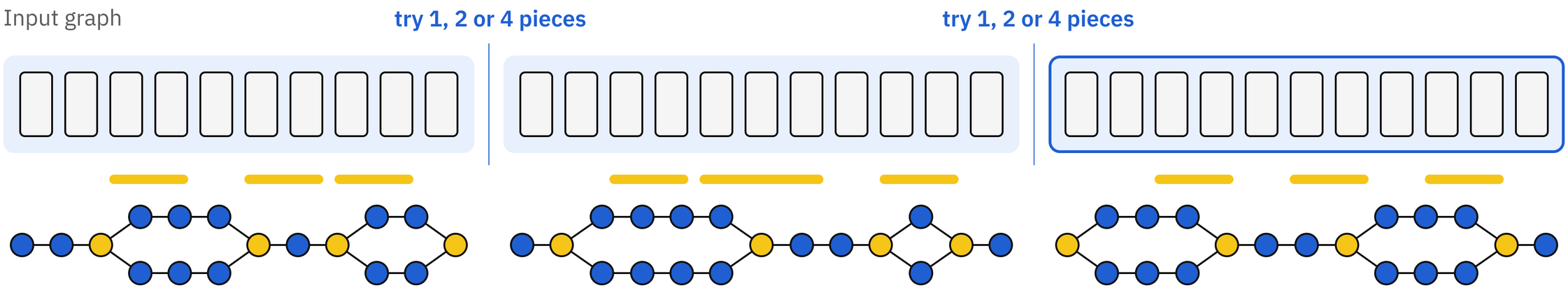
Level 2 runs on each part separately. The rewritten parts, joined, are the optimised graph.

→ NEXT: INSIDE ONE PART, LEVELS 2 AND 3

Illustration with made-up operators, matches, cuts and shapes. The minimum part size k = 10 and the cut criterion are from Section 5.3.

HOW IS OPTIMISATION PERFORMED? · LEVEL 1 WRAPS LEVEL 2

# Level 1: cut the model, then search each part




### The model

32 operators in a chain. Too many for one search: candidates grow exponentially.

### 1 · Match

Find where rule patterns occur in the whole graph (yellow).

### 2 · Cut

Parts of at least 10 operators, cutting the fewest matches. At each cut, try each way to partition the tensor.

### 3 · Search

Level 2 runs on each part separately. The rewritten parts, joined, are the optimised graph.

→ NEXT: INSIDE ONE PART, LEVELS 2 AND 3

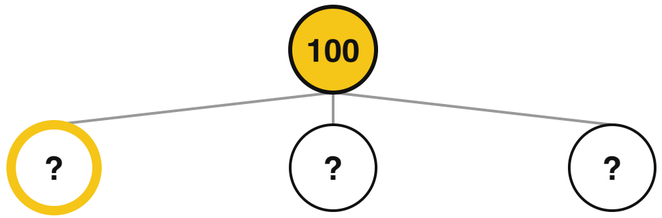
Illustration with made-up operators, matches, cuts and shapes. The minimum part size k = 10 and the cut criterion are from Section 5.3.

HOW IS OPTIMISATION PERFORMED? · LEVELS 2 AND 3, FOR ONE PART OF THE MODEL

# The placement runs inside every search step

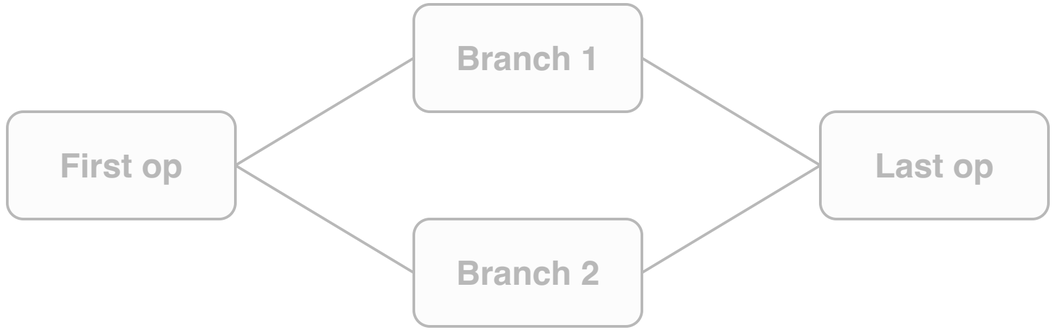
**A** · Take the best graph from the queue → **B** · Apply every rule at every place it fits → **C** · Estimate the cost of each new graph → **D** · Keep those within 1.05× of the best

## Level 2 · Substitution selection




Steps A and B: take the best graph and apply every rule. Three new graphs, cost unknown.

## Level 3 · Machine mapping (waiting)




Waits until the search needs the cost of a graph.

→ NEXT: DOES IT WORK? THE MEASURED RESULTS.

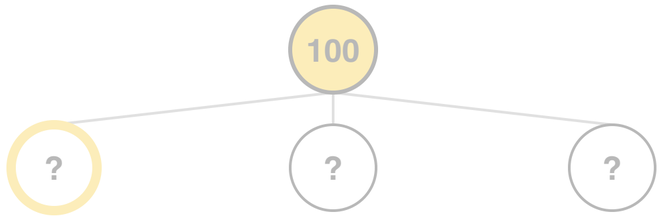
Level 1, graph splitting, runs this whole loop once per part of the model. Made-up numbers (ms per iteration); Sections 5.1 and 5.2. Yellow: best so far. Blue: in the queue. Dashed: pruned. “?”: cost not yet known.

HOW IS OPTIMISATION PERFORMED? · LEVELS 2 AND 3, FOR ONE PART OF THE MODEL

# The placement runs inside every search step

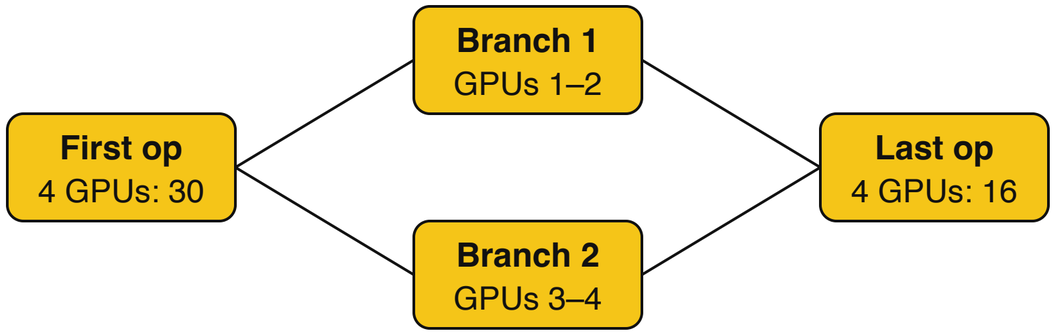
**A** · Take the best graph from the queue → **B** · Apply every rule at every place it fits → **C** · Estimate the cost of each new graph → **D** · Keep those within 1.05× of the best

## Level 2 · Substitution selection (waiting)




Steps A and B: take the best graph and apply every rule. Three new graphs, cost unknown.

## Level 3 · Machine mapping




Add the parts: 30 + 52 + 16 = 98. Back to the search.

→ NEXT: DOES IT WORK? THE MEASURED RESULTS.

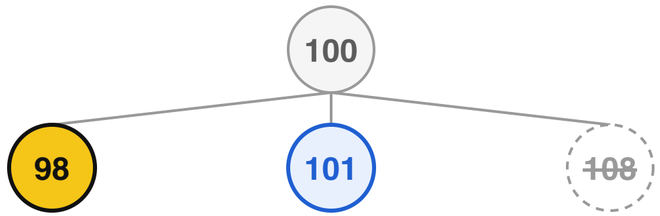
Level 1, graph splitting, runs this whole loop once per part of the model. Made-up numbers (ms per iteration); Sections 5.1 and 5.2. Yellow: best so far. Blue: in the queue. Dashed: pruned. “?”: cost not yet known.

HOW IS OPTIMISATION PERFORMED? · LEVELS 2 AND 3, FOR ONE PART OF THE MODEL

# The placement runs inside every search step

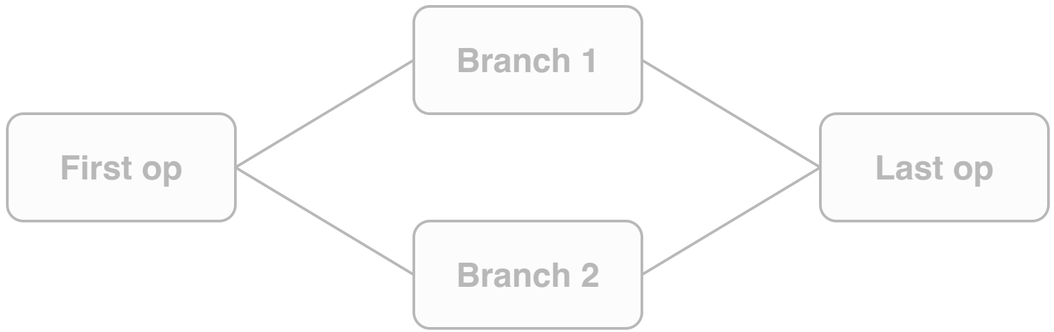
**A** · Take the best graph from the queue → **B** · Apply every rule at every place it fits → **C** · Estimate the cost of each new graph → **D** · Keep those within 1.05× of the best

## Level 2 · Substitution selection




Step D: 98 is the new best, 101 is within 5% and stays, 108 is pruned.

## Level 3 · Machine mapping (waiting)




Waits until the search needs the cost of a graph.

→ NEXT: DOES IT WORK? THE MEASURED RESULTS.

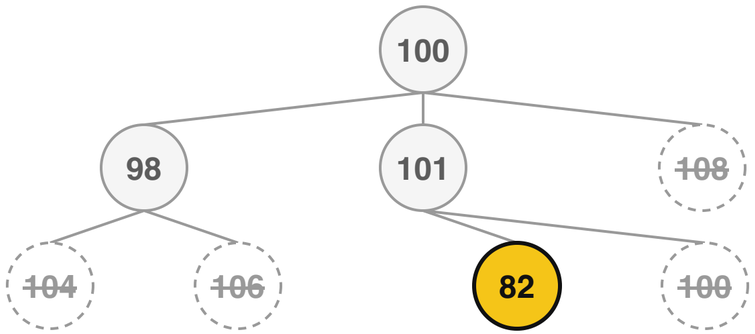
Level 1, graph splitting, runs this whole loop once per part of the model. Made-up numbers (ms per iteration); Sections 5.1 and 5.2. Yellow: best so far. Blue: in the queue. Dashed: pruned. “?”: cost not yet known.

HOW IS OPTIMISATION PERFORMED? · LEVELS 2 AND 3, FOR ONE PART OF THE MODEL

# The placement runs inside every search step

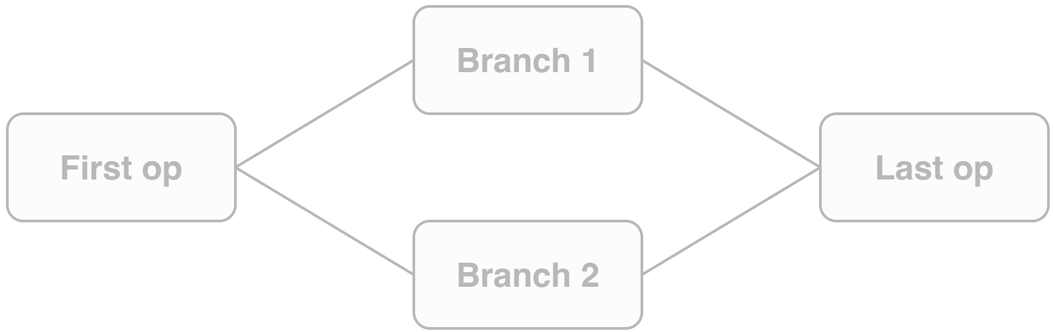
**A** · Take the best graph from the queue → **B** · Apply every rule at every place it fits → **C** · Estimate the cost of each new graph → **D** · Keep those within 1.05× of the best

## Level 2 · Substitution selection




82 is the new best. The slightly worse graph led to it.

## Level 3 · Machine mapping (waiting)



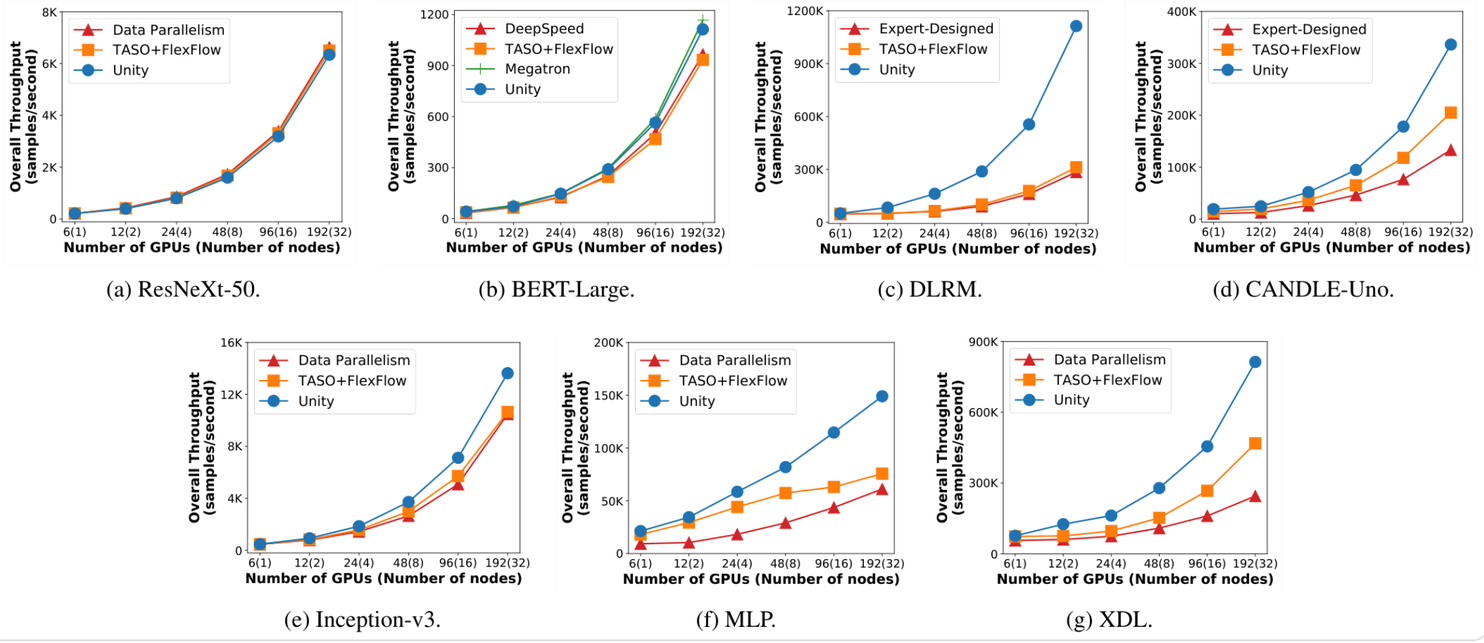

Waits until the search needs the cost of a graph.

→ NEXT: DOES IT WORK? THE MEASURED RESULTS.

Level 1, graph splitting, runs this whole loop once per part of the model. Made-up numbers (ms per iteration); Sections 5.1 and 5.2. Yellow: best so far. Blue: in the queue. Dashed: pruned. “?”: cost not yet known.

HOW IS PERFORMANCE EVALUATED? · MEASURED TRAINING THROUGHPUT

# Unity matches experts and beats them on mixed models




(a) ResNeXt-50. (b) BERT-Large. (c) DLRM. (d) CANDLE-Uno.

(e) Inception-v3. (f) MLP. (g) XDL.

**3.6×**

at best, on DLRM: over the expert strategy and over TASO + FlexFlow

**1.6×**

at best, on CANDLE-Uno

**≈ 1×**

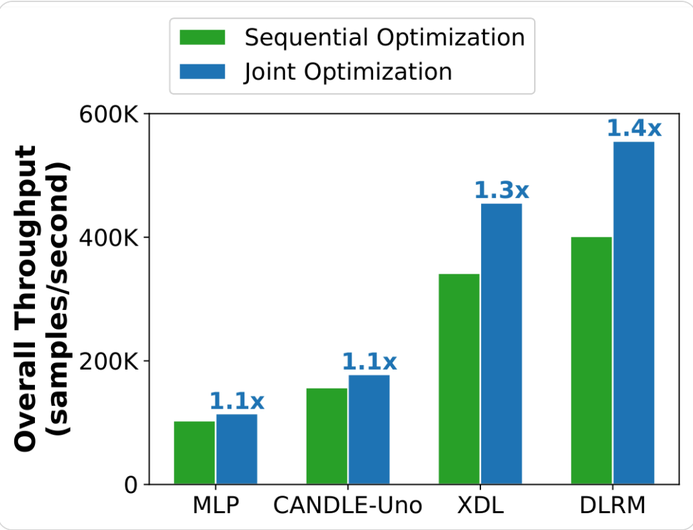
vs Megatron on BERT-Large: the hand-designed strategy, rediscovered by a search of under 10 min

→ NEXT: HOW MUCH OF THIS COMES FROM THE JOINT SEARCH?

Figure 14 of the paper. Seven DNNs, up to 192 GPUs on 32 nodes of the Summit supercomputer; each point averages 1,000 iterations. Search: under 10 min for six models, under 20 min for Inception-v3.

WHICH EVIDENCE SUPPORTS WHICH CLAIM?

# The joint search alone accounts for up to 1.4×




**< 20 min**

search time, against training runs of hours or days

END TO END · FIGURE 14 · UP TO 192 GPUS · EVERYTHING DIFFERS

| Run | Joint search | More parallelism | Scalable search | Speed |
|---|---|---|---|---|
| TASO + FlexFlow | – off | – off | – off | 1× |
| Unity | ✓ on | ✓ on | ✓ on | up to 3.6× |

ABLATION · FIGURE 15B · 96 GPUS · ONLY THE JOINT SEARCH DIFFERS

| Run | Joint search | More parallelism | Scalable search | Speed |
|---|---|---|---|---|
| Unity, sequential | – off | ✓ on | ✓ on | 1× |
| Unity, joint | ✓ on | ✓ on | ✓ on | up to 1.4× |

Open point: candidates are ranked by an estimated cost, and the paper does not report how accurate that estimate is.

→ NEXT: WHAT THE PAPER LEAVES OPEN

Figures 14 and 15b, Sections 6.2 and 6.4 of the paper. Bars: sequential (green) vs joint (blue) on MLP, CANDLE-Uno, XDL and DLRM.

ASSESSMENT

# A strong idea, with limits in method and evidence

## Strengths

- **One representation.** Rewrites and parallelisation become the same kind of move, so one search covers both.
- **Proven and practical.** Verified substitutions; matches or beats experts on seven models after minutes of search.

## Limits of the method

- ⚠ **Memory is ignored.** A chosen strategy may not fit on the GPUs.
- ⚠ **Unvalidated cost estimate.** Every choice rests on it; its accuracy is not reported.
- ⚠ **Heuristic, restricted search.** No optimality guarantee; the 5% rule is never varied.

## Limits of the evidence

- ⚠ **Own baselines.** TASO + FlexFlow are the authors' 2019 systems; other tools are not run.
- ⚠ **3.6× is not broken down.** "Nearly half from the joint search" is not derived.
- ⚠ **Narrow claims and tests.** "First" means first automatic; one type of GPU only.

**For discussion: could a learned cost model or policy replace the estimate and the 5% rule?**

→ NEXT: THE TAKEAWAY

TAKEAWAY

# Put the choices in one space, then search it

| WHAT IS OPTIMISED | REPRESENTATION | HOW | EVALUATED BY |
| --- | --- | --- | --- |
| **Rewrites + parallelism + placement** | **Parallel computation graph** | **Verified substitutions + hierarchical search** | **Estimated cost, then measured throughput** |

**RESULT**

Up to 3.6× over existing systems; up to 1.4× from searching jointly

**WHERE LEARNING COULD ENTER**

A learned cost model, or a learned policy to guide the search

**When is an explicit representation plus structured search enough, and when should we learn a policy?**

Questions?

REFERENCES

# Sources

MAIN PAPER · ALL FIGURES AND NUMBERS

**Unity: Accelerating DNN Training Through Joint Optimization of Algebraic Transformations and Parallelization.** C. Unger, Z. Jia, W. Wu, S. Lin, M. Baines, et al., A. Aiken. OSDI 2022.

ALSO MENTIONED

**Device placement with RL** · Mirhoseini et al., ICML 2017

**Placeto** · Addanki et al., arXiv 2019

**KernelCraft** · Nie et al., ICML 2026

**TENSAT** · Yang et al., MLSys 2021

**TASO** · Jia et al., SOSP 2019

**FlexFlow** · Jia, Zaharia, Aiken, SysML 2019

**Megatron-LM** · Shoeybi et al., 2019

**X-RLflow** · He, Parker, Yoneki, MLSys 2023

TAKEAWAY

# Put the choices in one space, then search it

| WHAT IS OPTIMISED | REPRESENTATION | HOW | EVALUATED BY |
| --- | --- | --- | --- |
| **Rewrites + parallelism + placement** | **Parallel computation graph** | **Verified substitutions + hierarchical search** | **Estimated cost, then measured throughput** |

**RESULT**

Up to 3.6× over existing systems; up to 1.4× from searching jointly

**WHERE LEARNING COULD ENTER**

A learned cost model, or a learned policy to guide the search

**When is an explicit representation plus structured search enough, and when should we learn a policy?**

Questions?