THE UNIVERSITY of EDINBURGH
informatics

EPSRC Centre for Doctoral Training in
Pervasive Parallelism

Mapping Parallelism in a Functional IR through Constraint Satisfaction

A Case Study on Convolution for Mobile GPUs

Artifact:
https://gitlab.com/naummo/liftpar-cc-2022-artifact

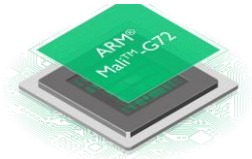
Naums Mogers, Lu Li,
Valentin Radu, Christophe Dubach

April 6th, 2022

Parallel Architectures

Parallel architectures are hard to optimize for

©ARM



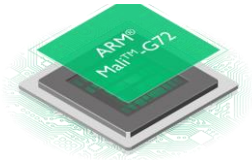
©NVIDIA

©Xilinx

2

Parallel Architectures

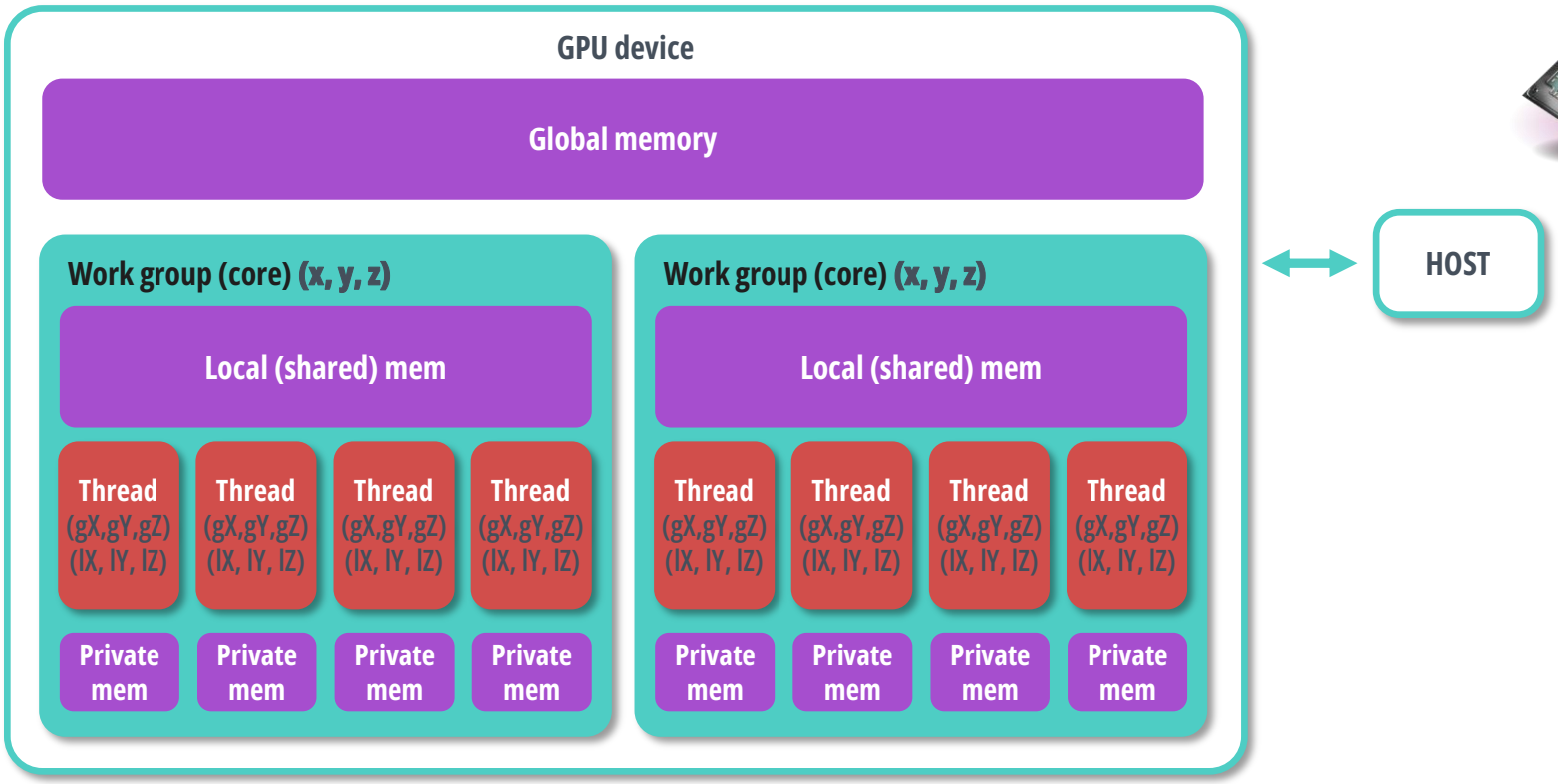
©ARM



©NVIDIA

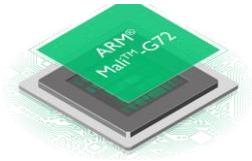
©Xilinx

Parallel architectures are hard to optimize for



Parallel Architectures

©ARM



©NVIDIA

©Xilinx

Parallel architectures are hard to optimize for

Large implementation space

Work group (core) (x, y, z)

Local (shared) mem

Thread (gX,gY,gZ) (lX, lY, lZ)

Private mem

HOST

Automatic code generators and optimizers are needed

Parallelization Is Hard

```
1  for i in 1 to I do
2    for j in 1 to J do
3      for k in 1 to K do
4        shared_buf[j][k] = f( input[i][j][k] );
5
6    for k in 1 to K do
7      for j in 1 to J do
8        output[i][j][k] = g( shared_buf[j][k] );
```

Transposed!

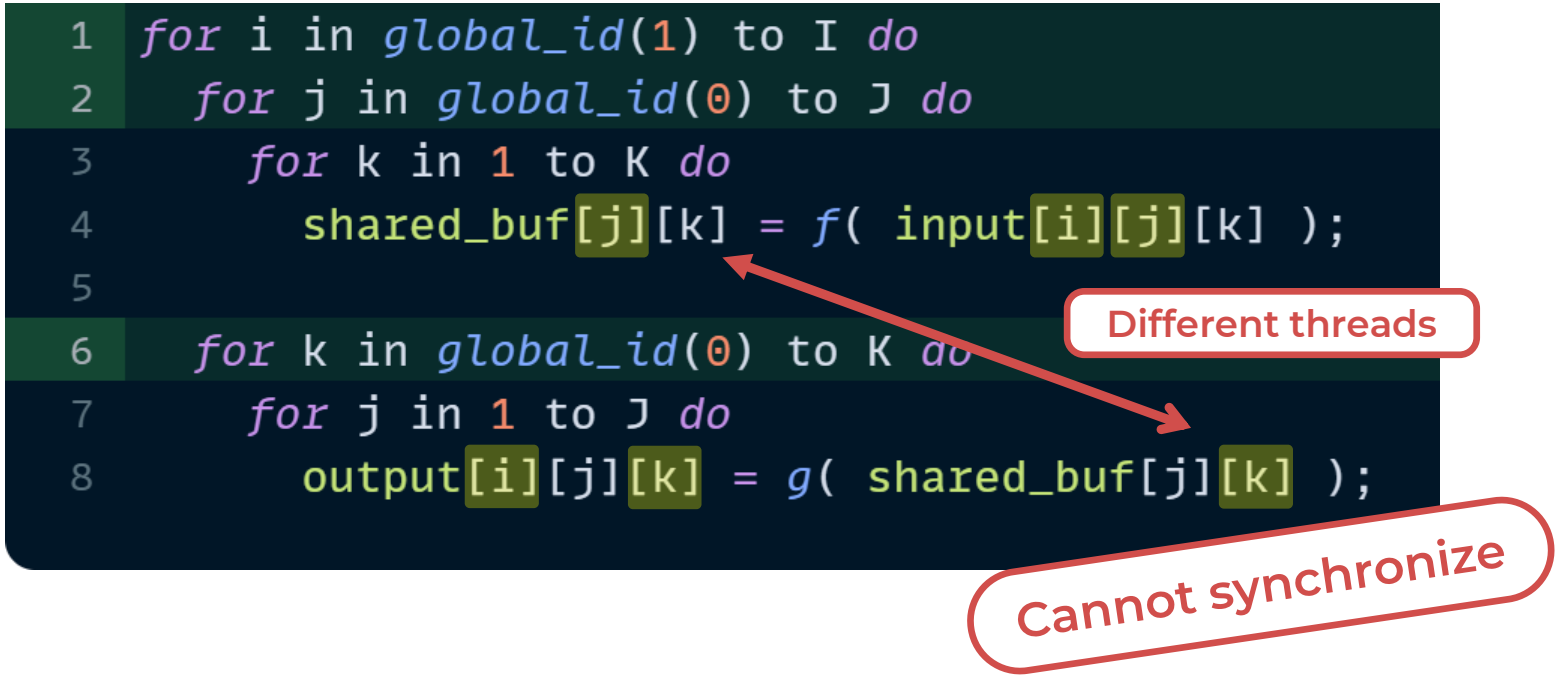
Parallelization Is Hard

```
1  for i in global_id(0) to I do
2    for j in 1 to J do
3      for k in 1 to K do
4        shared_buf[j][k] = f( input[i][j][k] );
5
6    for k in 1 to K do
7      for j in 1 to J do
8        output[i][j][k] = g( shared_buf[j][k] );
```

Under-saturated cores

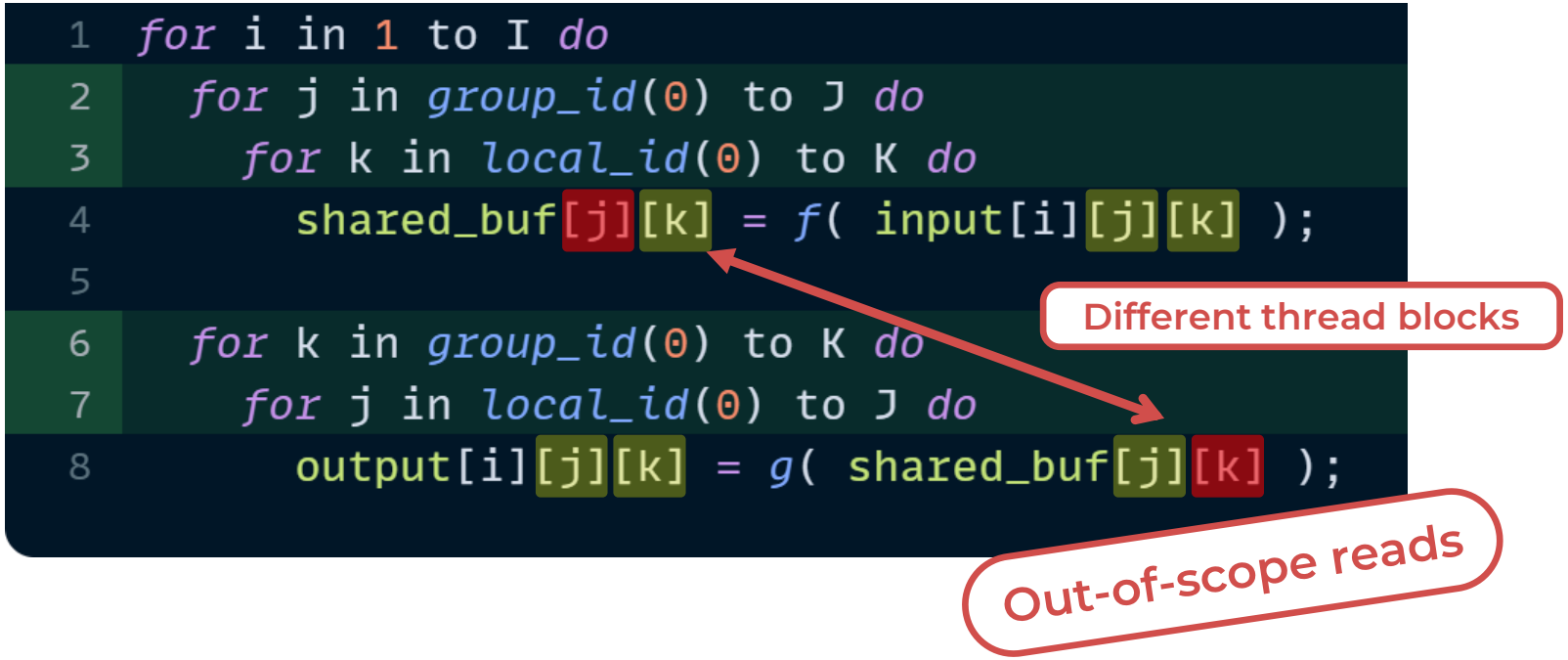
Parallelization Is Hard

```
1  for i in global_id(1) to I do
2    for j in global_id(0) to J do
3      for k in 1 to K do
4        shared_buf[j][k] = f( input[i][j][k] );
5
6    for k in global_id(0) to K do
7      for j in 1 to J do
8        output[i][j][k] = g( shared_buf[j][k] );
```



Parallelization Is Hard

```
1  for i in 1 to I do
2    for j in group_id(0) to J do
3      for k in local_id(0) to K do
4        shared_buf[j][k] = f( input[i][j][k] );
5
6    for k in group_id(0) to K do
7      for j in local_id(0) to J do
8        output[i][j][k] = g( shared_buf[j][k] );
```



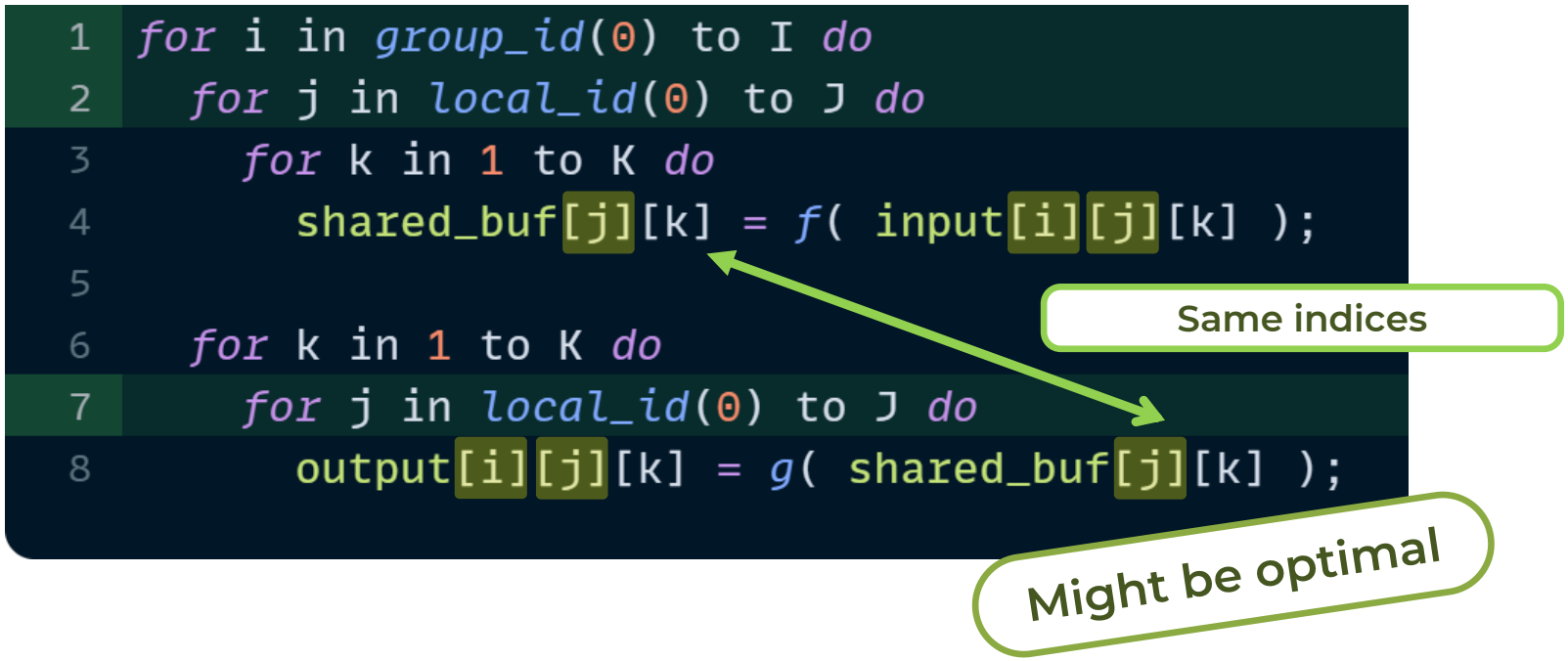
Parallelization Is Hard

```
1  for i in group_id(0) to I do
2    for j in local_id(0) to J do
3      for k in 1 to K do
4        shared_buf[j][k] = f( input[i][j][k] );
5    barrier();
6    for k in local_id(0) to K do
7      for j in 1 to J do
8        output[i][j][k] = g( shared_buf[j][k] );
```

Synchronization overhead

Parallelization Is Hard

```
1  for i in group_id(0) to I do
2    for j in local_id(0) to J do
3      for k in 1 to K do
4        shared_buf[j][k] = f( input[i][j][k] );
5
6    for k in 1 to K do
7      for j in local_id(0) to J do
8        output[i][j][k] = g( shared_buf[j][k] );
```



Parallelization: Approaches

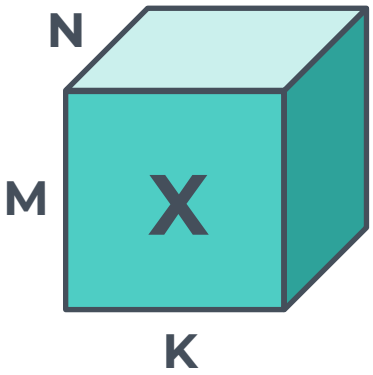
- User-provided schedules
- Polyhedral compilation + heuristics
- Functional IR *(Accelerate, Futhark)* + heuristics

- Proposed approach:
  - Functional IR
  - Generate parallelization constraints
  - Explore the design space automatically
    - **Avoid evaluating invalid parallel mappings**

Example

```
1  map^A (
2
3
4
5  ) $ (x: [[[T]_K]_M]_N)
```

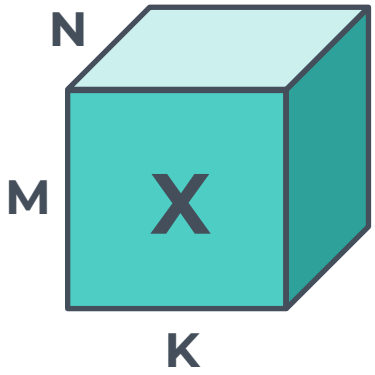
Application



Example

```
1  map^A (
2
3     map^C (
4                          map^D (toPrivate(id)))
5  ) $ (x: [[[T]_K]_M]_N)
```

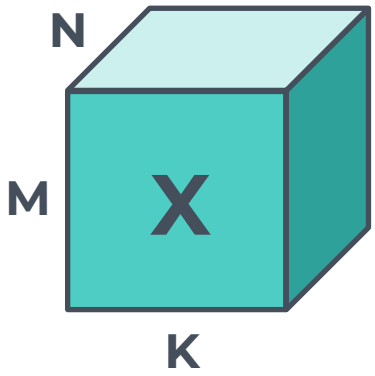
Identity function (copy)



Example

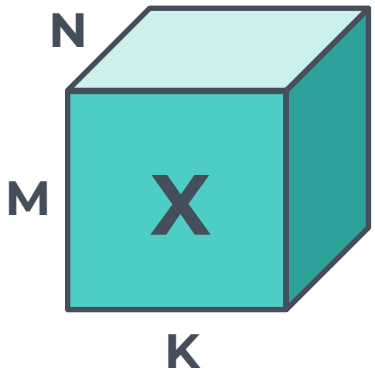
```
1  map^A (
2
3    map^C (
4      toLocal(f) ∘ map^D (toPrivate(id))
5  ) $ (X: [[[T]_K]_M]_N)
```

Composition

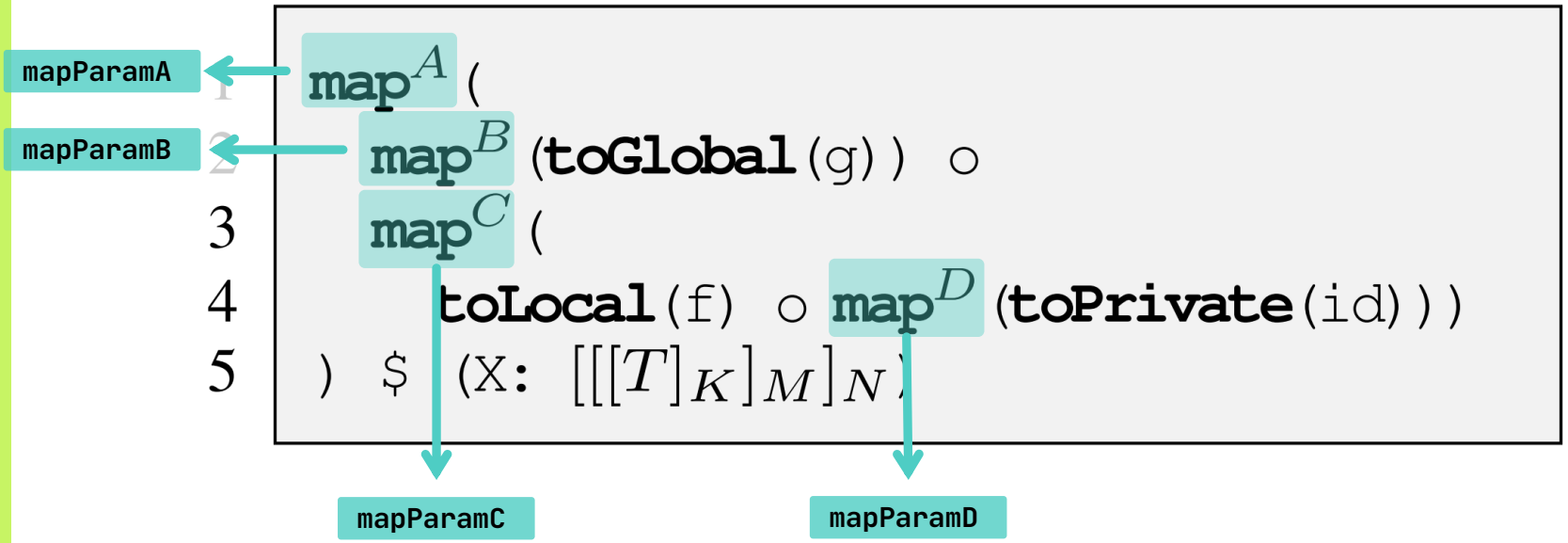


Example

```
1  map^A (
2    map^B (toGlobal(g)) ∘
3    map^C (
4      toLocal(f) ∘ map^D (toPrivate(id)))
5  ) $ (x: [[[T]_K]_M]_N)
```



Parallelization:
Scheduling Parameters



- Associate a parameter with each `map`
- Encode scheduling choices as integers
- Model parallelization restrictions as integer constraints

Parallelization:
Encoding of Choices

Code value	Map transformation
0	`mapSeq`
1	Fused with the outer `map`
10, 11, 12	`mapLcl` in dimension 0, 1 or 2 respectively
20, 21, 22	`mapWrg` in dimension 0, 1 or 2 respectively
30, 31, 32	`mapGlb` in dimension 0, 1 or 2 respectively

Constraint:
Private Memory Scope

IF: accesses private memory
THEN: cannot be parallel

```
1  map^A (
2    map^B (toGlobal(g)) ∘
3    map^C (
4      toLocal(f) ∘ map^D (toPrivate(id))
5  ) $ (x: [[[T]_K]_M]_N)
```

Private memory access constraint. `maps` that consume or produce private memory cannot be parallelized because private memory is restricted in scope to a single thread.

Constraint:
Private Memory Scope

IF: accesses private memory
THEN: cannot be parallel

```
1  map^A (
2    map^B (toGlobal(g)) ∘
3    map^C (
4      toLocal(f) ∘ map^D (toPrivate(id))
5  ) $ (X: [[[T]_K]_M]_N)
```

Private memory access constraint. `maps` that consume or produce private memory cannot be parallelized because private memory is restricted in scope to a single thread.

```
∀m : m.usesPrivateMemory
```

GEN CONSTRAINT: `mapEncoding(m)/10 < 1`

Code value	Map transformation
0	`mapSeq`
1	Fused with the outer `map`
10, 11, 12	`mapLcl` in dimension 0, 1 or 2
20, 21, 22	`mapWrg` in dimension 0, 1 or 2
30, 31, 32	`mapGlb` in dimension 0, 1 or 2

Constraint:
Private Memory Scope

```
1  map^A (
2    map^B (toGlobal(g)) ∘
3    map^C (
4      toLocal(f) ∘ map^D (toPrivate(id))
5  ) $ (X: [[[T]_K]_M]_N)
```

$(\texttt{mapEncoding}(\texttt{mapParamD})/10 < 1)$

Constraint:
Shared Memory Scope

```
1  map^A (
2     map^B (toGlobal(g)) ∘
3     map^C (
4        toLocal(f) ∘ map^D (toPrivate(id)))
5  ) $ (X: [[[T]_K]_M]_N)
```

$(\texttt{mapEncoding}(\texttt{mapParamD})/10 < 1)$

map^C is local or sequential ∧ **map^A** is a mapWrg

Constraint:
Shared Memory Scope

```
1  map^A (
2    map^B (toGlobal(g)) ∘
3    map^C (
4      toLocal(f) ∘ map^D (toPrivate(id)))
5  ) $ (X: [[[T]_K]_M]_N)
```

$(\texttt{mapEncoding}(\texttt{mapParamD})/10 < 1)$

$(\texttt{mapEncoding}(\texttt{mapParamC})/10 < 2) \wedge \texttt{mapEncoding}(\texttt{mapParamA}) = 20 + \texttt{w}$

Constraint:
Shared Memory Scope

```
1  map^A (
2    map^B (toGlobal(g)) ∘
3    map^C (
4      toLocal(f) ∘ map^D (toPrivate(id)))
5  ) $ (X: [[[T]_K]_M]_N)
```

$$(\texttt{mapEncoding}(\texttt{mapParamD})/10 < 1)$$
$$(\texttt{mapEncoding}(\texttt{mapParamC})/10 < 2) \wedge \texttt{mapEncoding}(\texttt{mapParamA}) = 20 + \texttt{w}$$
$$(\texttt{mapEncoding}(\texttt{mapParamB})/10 < 2) \wedge \texttt{mapEncoding}(\texttt{mapParamA}) = 20 + \texttt{w}$$

Constraint:
Hierarchical Parallelism

```
1  map^A (
2     map^B (toGlobal(g)) ∘
3     map^C (
4        toLocal(f) ∘ map^D (toPrivate(id)))
5  ) $ (x: [[[T]_K]_M]_N)
```

$$(\texttt{mapEncoding(mapParamD)}/10 < 1)$$

$$(\texttt{mapEncoding(mapParamC)}/10 < 2) \wedge \texttt{mapEncoding(mapParamA)} = 20 + \texttt{w}$$

$$(\texttt{mapEncoding(mapParamB)}/10 < 2) \wedge \texttt{mapEncoding(mapParamA)} = 20 + \texttt{w}$$

map^A is not parallel $\vee$ **map^A** and **map^B** are not parallelized in the same way

Constraint:
Hierarchical Parallelism

```
1  map^A (
2    map^B (toGlobal(g)) ∘
3    map^C (
4      toLocal(f) ∘ map^D (toPrivate(id)))
5  ) $ (x: [[[T]_K]_M]_N)
```

$$(\texttt{mapEncoding}(\texttt{mapParamD})/10 < 1)$$
$$(\texttt{mapEncoding}(\texttt{mapParamC})/10 < 2) \wedge \texttt{mapEncoding}(\texttt{mapParamA}) = 20 + \texttt{w}$$
$$(\texttt{mapEncoding}(\texttt{mapParamB})/10 < 2) \wedge \texttt{mapEncoding}(\texttt{mapParamA}) = 20 + \texttt{w}$$
$$(\texttt{mapEncoding}(\texttt{mapParamA})/10 < 1) \vee (\texttt{mapEncoding}(\texttt{mapParamA}) \neq \texttt{mapEncoding}(\texttt{mapParamB}))$$
$$(\texttt{mapEncoding}(\texttt{mapParamA})/10 < 1) \vee (\texttt{mapEncoding}(\texttt{mapParamA}) \neq \texttt{mapEncoding}(\texttt{mapParamC}))$$
$$\neg((\texttt{mapEncoding}(\texttt{mapParamA})/10 = 1) \wedge (\texttt{mapEncoding}(\texttt{mapParamB})/10 = 2) \wedge (\texttt{mapEncoding}(\texttt{mapParamB})\,\%10 = \texttt{mapEncoding}(\texttt{mapParamA})\,\%10))$$

+10 more hierarchical parallelism constraints

Constraint Satisfaction

```
1  map^A (
2    map^B (toGlobal(g)) ∘
3    map^C (
4      toLocal(f) ∘ map^D (toPrivate(id)
5  ) $ (x: [[[T]_K]_M]_N)
```

```
mapParamA = 20
mapParamB = 10
mapParamC = 10
mapParamD = 0
```

$$(\texttt{mapEncoding}_{\texttt{0}}(\texttt{mapParamD})\ /10 < 1)$$
$$(\texttt{mapEncoding}_{\texttt{10}}(\texttt{mapParamC})\ /10 < 2)\ \wedge\ \texttt{mapEncoding}_{\texttt{20}}(\texttt{mapParamA}) = 20 + \texttt{w}$$
$$(\texttt{mapEncoding}_{\texttt{10}}(\texttt{mapParamB})\ /10 < 2)\ \wedge\ \texttt{mapEncoding}_{\texttt{20}}(\texttt{mapParamA}) = 20 + \texttt{w}$$
$$(\texttt{mapEncoding}_{\texttt{20}}(\texttt{mapParamA})\ /10 < 1)\ \vee\ (\texttt{mapEncoding}_{\texttt{20}}(\texttt{mapParamA}) \neq \texttt{mapEncoding}_{\texttt{10}}(\texttt{mapParamB}))$$
$$(\texttt{mapEncoding}_{\texttt{20}}(\texttt{mapParamA})\ /10 < 1)\ \vee\ (\texttt{mapEncoding}_{\texttt{20}}(\texttt{mapParamA}) \neq \texttt{mapEncoding}_{\texttt{10}}(\texttt{mapParamC}))$$
$$\neg((\texttt{mapEncoding}_{\texttt{20}}(\texttt{mapParamA})\ /10 = 1)\ \wedge\ (\texttt{mapEncoding}_{\texttt{10}}(\texttt{mapParamB})\ /10 = 2)\ \wedge$$
$$(\texttt{mapEncoding}_{\texttt{10}}(\texttt{mapParamB})\ \%10 = \texttt{mapEncoding}_{\texttt{20}}(\texttt{mapParamA})\ \%10))$$

+10 more hierarchical parallelism constraints

Constraint Satisfaction

```
1  mapWrg(0)(
2    mapLcl(0)(toGlobal(g)) ∘
3    mapLcl(0)(
4      toLocal(f) ∘ mapSeq(toPrivate(i
5  ) $ (X: [[[T]_K]_M]_N)
```

```
mapParamA = 20
mapParamB = 10
mapParamC = 10
mapParamD = 0
```

$$
\begin{aligned}
&(\text{mapEncoding}_{0}(\text{mapParamD})/10 < 1)\\
&(\text{mapEncoding}_{10}(\text{mapParamC})/10 < 2) \wedge \text{mapEncoding}_{20}(\text{mapParamA}) = 20 + \text{w}\\
&(\text{mapEncoding}_{10}(\text{mapParamB})/10 < 2) \wedge \text{mapEncoding}_{20}(\text{mapParamA}) = 20 + \text{w}\\
&(\text{mapEncoding}_{20}(\text{mapParamA})/10 < 1) \vee (\text{mapEncoding}_{20}(\text{mapParamA}) \neq \text{mapEncoding}_{10}(\text{mapParamB}))\\
&(\text{mapEncoding}_{20}(\text{mapParamA})/10 < 1) \vee (\text{mapEncoding}_{20}(\text{mapParamA}) \neq \text{mapEncoding}_{10}(\text{mapParamC}))\\
&\neg((\text{mapEncoding}_{20}(\text{mapParamA})/10 = 1) \wedge (\text{mapEncoding}_{10}(\text{mapParamB})/10 = 2) \wedge\\
&\quad(\text{mapEncoding}_{10}(\text{mapParamB})\%10 = \text{mapEncoding}_{20}(\text{mapParamA})\%10))
\end{aligned}
$$

+10 more hierarchical parallelism constraints

Heuristics

Sequential Map Fusion Heuristic

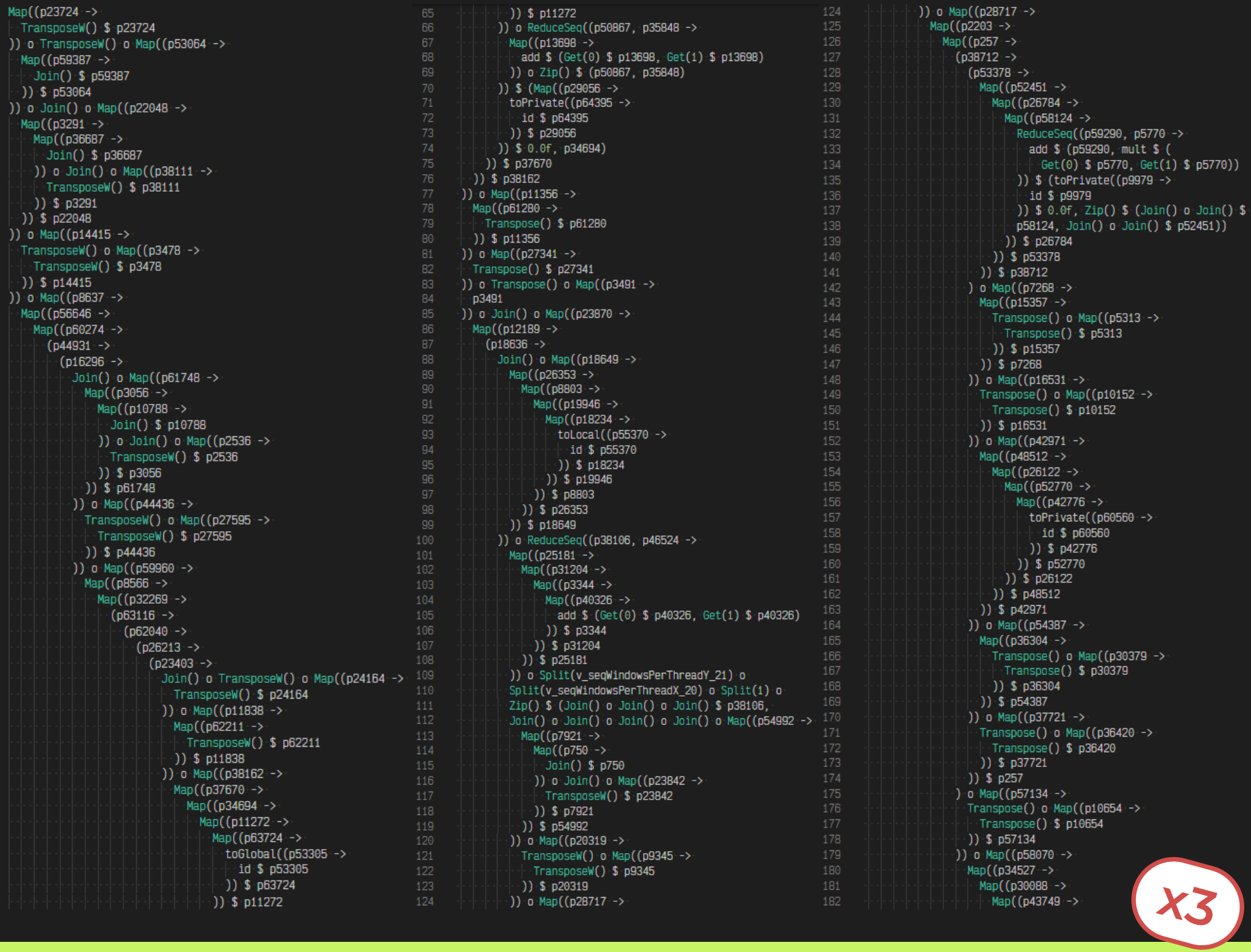
Perfectly nested sequential maps can always be fused to reduce search space

$$\forall \texttt{Chain} \in \texttt{MapNestingChains},\ \forall \texttt{m1} \in \texttt{Chain},\ \forall \texttt{m2} \in \texttt{Chain},$$
$$\texttt{m2.perfectlyNestedIn(m1)}$$
$$\textbf{GEN CONSTRAINT: } \neg((\texttt{mapEncoding(m1)} = 0) \wedge (\texttt{mapEncoding(m2)} = 0))$$

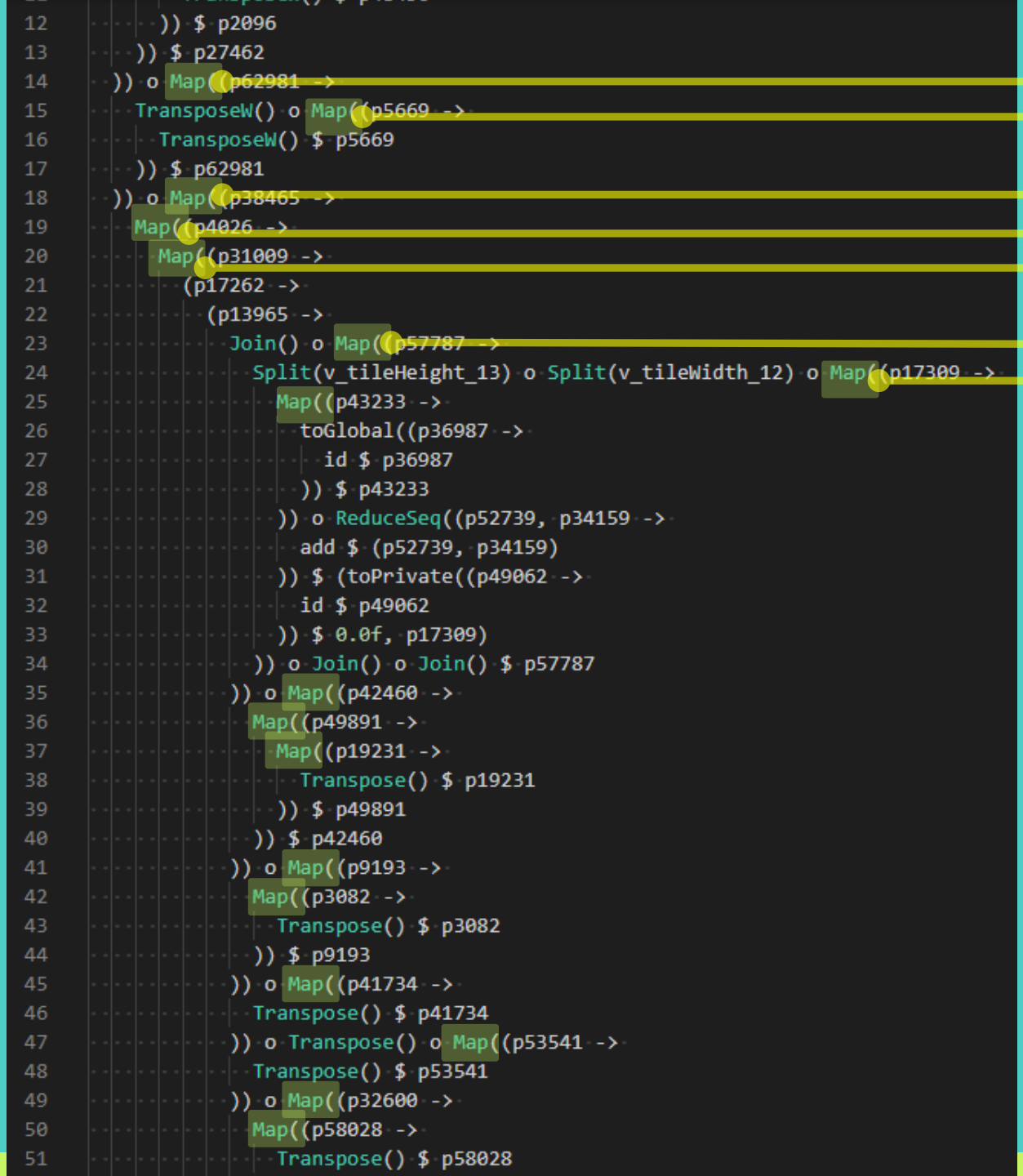
Code value	Map transformation
0	mapSeq
1	Fused with the outer map
10, 11, 12	mapLcl in dimension 0, 1 or 2 respectively
20, 21, 22	mapWrg in dimension 0, 1 or 2 respectively
30, 31, 32	mapGlb in dimension 0, 1 or 2 respectively

Convolution in Lift

```
1  def conv( in: [[[T]_inChs]_inW]_inH ,
2            ks: [[[[T]_inChs]_kerW]_kerH]_outChs ,
3            kerStepX: int , kerStepY: int
4          ) : [[[T]_outChs]_outW]_outH =
5   mapND_2(slideWin: [[[T]_inChs]_kerW]_kerH ⇒
6     map(singleK: [[[T]_inChs]_kerW]_kerH ⇒
7       reduce(+, 0, map(*,
8         joinND_2(zipND_3(slideWin, singleK)))),
9       ks), slideND_2(kerH, kerStepY, kerW, kerStepX, in))
```


x3

```
    )) $ p2096
  )) $ p27462
)) o Map((p62981 ->
  TransposeW() o Map((p5669 ->
    TransposeW() $ p5669
  )) $ p62981
)) o Map((p38465 ->
  Map((p4026 ->
    Map((p31009 ->
      (p17262 ->
        (p13965 ->
          Join() o Map((p57787 ->
            Split(v_tileHeight_13) o Split(v_tileWidth_12) o Map((p17309 ->
              Map((p43233 ->
                toGlobal((p36987 ->
                  id $ p36987
                )) $ p43233
              )) o ReduceSeq((p52739, p34159 ->
                add $ (p52739, p34159)
              )) $ (toPrivate((p49062 ->
                id $ p49062
              )) $ 0.0f, p17309)
            )) o Join() o Join() $ p57787
          )) o Map((p42460 ->
            Map((p49891 ->
              Map((p19231 ->
                Transpose() $ p19231
              )) $ p49891
            )) $ p42460
          )) o Map((p9193 ->
            Map((p3082 ->
              Transpose() $ p3082
            )) $ p9193
          )) o Map((p41734 ->
            Transpose() $ p41734
          )) o Transpose() o Map((p53541 ->
            Transpose() $ p53541
          )) o Map((p32600 ->
            Map((p58028 ->
              Transpose() $ p58028
```



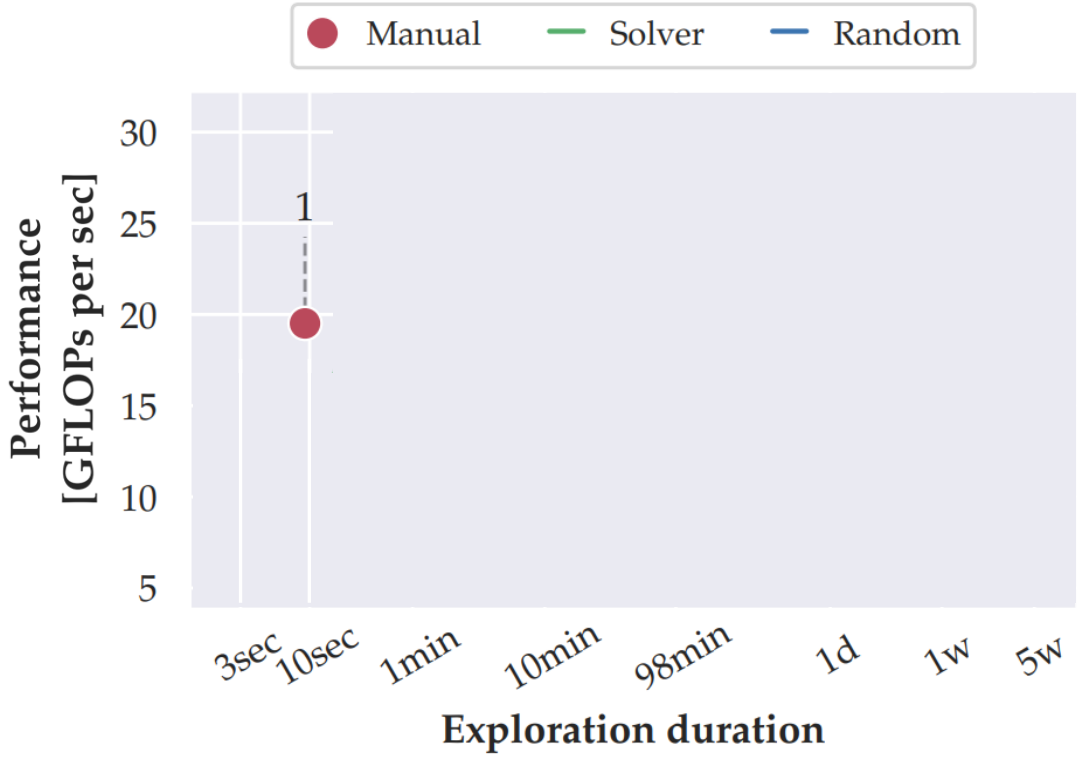

```
12        )) $ p2096
13      )) $ p27462
14    )) o Map((p62981 ->
15      TransposeW() o Map((p5669 ->
16        TransposeW() $ p5669
17      )) $ p62981
18    )) o Map((p38465 ->
19      Map((p4026 ->
20        Map((p31009 ->
21          (p17262 ->
22            (p13965 ->
23              Join() o Map((p57787 ->
24                Split(v_tileHeight_13) o Split(v_tileWidth_12) o Map((p17309 ->
25                  Map((p43233 ->
26                    toGlobal((p36987 ->
27                      id $ p36987
28                    )) $ p43233
29                  )) o ReduceSeq((p52739, p34159 ->
30                    add $ (p52739, p34159)
31                  )) $ (toPrivate((p49062 ->
32                    id $ p49062
33                  )) $ 0.0f, p17309)
34                )) o Join() o Join() $ p57787
35              )) o Map((p42460 ->
36                Map((p49891 ->
37                  Map((p19231 ->
38                    Transpose() $ p19231
39                  )) $ p49891
40                )) $ p42460
41              )) o Map((p9193 ->
42                Map((p3082 ->
43                  Transpose() $ p3082
44                )) $ p9193
45              )) o Map((p41734 ->
46                Transpose() $ p41734
47              )) o Transpose() o Map((p53541 ->
48                Transpose() $ p53541
49              )) o Map((p32600 ->
50                Map((p58028 ->
51                  Transpose() $ p58028
```


32

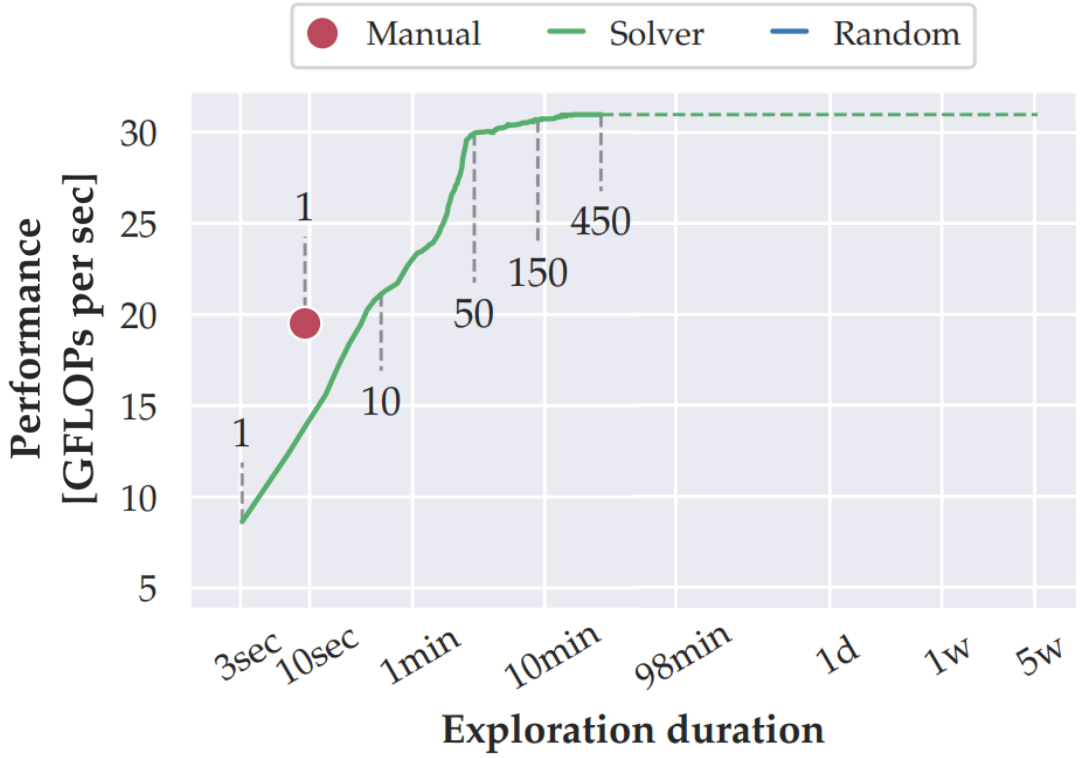
```
12        )) $ p2096
13      )) $ p27462
14    )) o Map((p62981 ->
15      TransposeW() o Map((p5669 ->
16        TransposeW() $ p5669
17      )) $ p62981
18    )) o MapWrg(2,(p38465 ->
19      MapWrg(0,(p4026 ->
20        MapSeq((p31009 ->
21          (p17262 ->
22            (p13965 ->
23              Join() o MapLcl(2,(p57787 ->
24                Split(v_tileHeight_13) o Split(v_tileWidth_12) o MapLcl(0,(p17309
25                  MapSeq((p43233 ->
26                    toGlobal((p36987 ->
27                      id $ p36987
28                    )) $ p43233
29                  )) o ReduceSeq((p52739, p34159 ->
30                    add $ (p52739, p34159)
31                  )) $ (toPrivate((p49062 ->
32                    id $ p49062
33                  )) $ 0.0f, p17309)
34                )) o Join() o Join() $ p57787
35              )) o Map((p42460 ->
36                Map((p49891 ->
37                  Map((p19231 ->
38                    Transpose() $ p19231
39                  )) $ p49891
40                )) $ p42460
41              )) o Map((p9193 ->
42                Map((p3082 ->
43                  Transpose() $ p3082
44                )) $ p9193
45              )) o Map((p41734 ->
46                Transpose() $ p41734
47              )) o Transpose() o Map((p53541 ->
48                Transpose() $ p53541
49              )) o Map((p32600 ->
50                Map((p58028 ->
51                  Transpose() $ p58028
```


33

Results: Exploration Efficiency

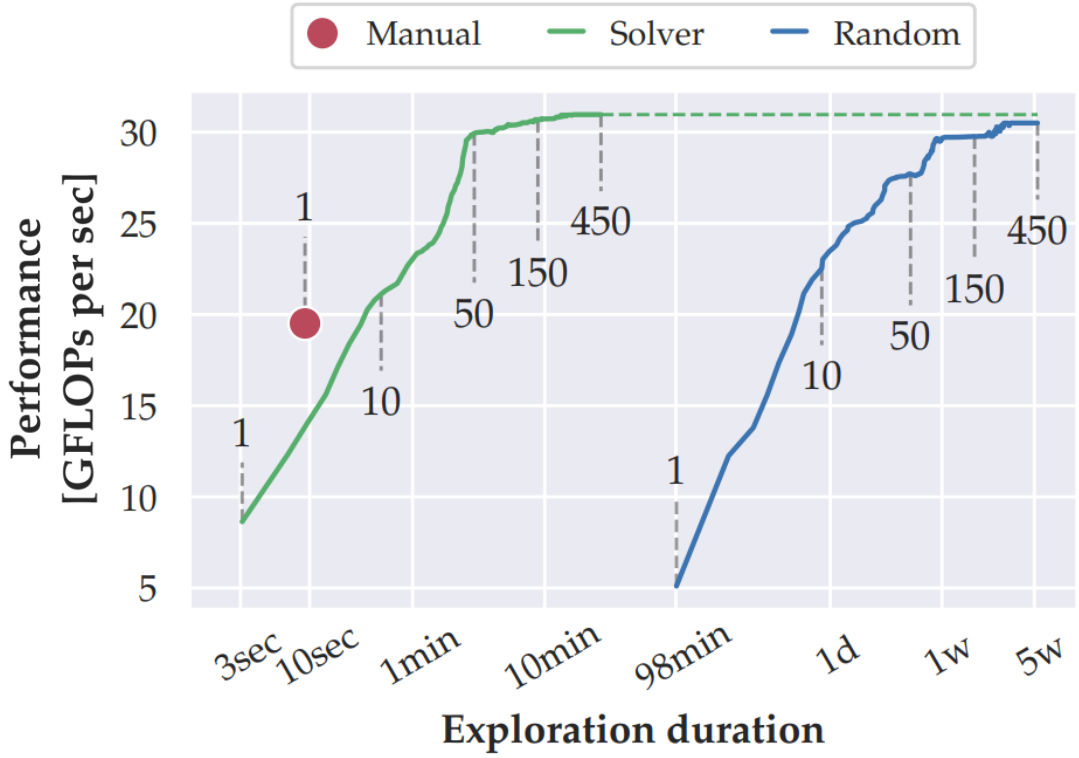


Results: Exploration Efficiency



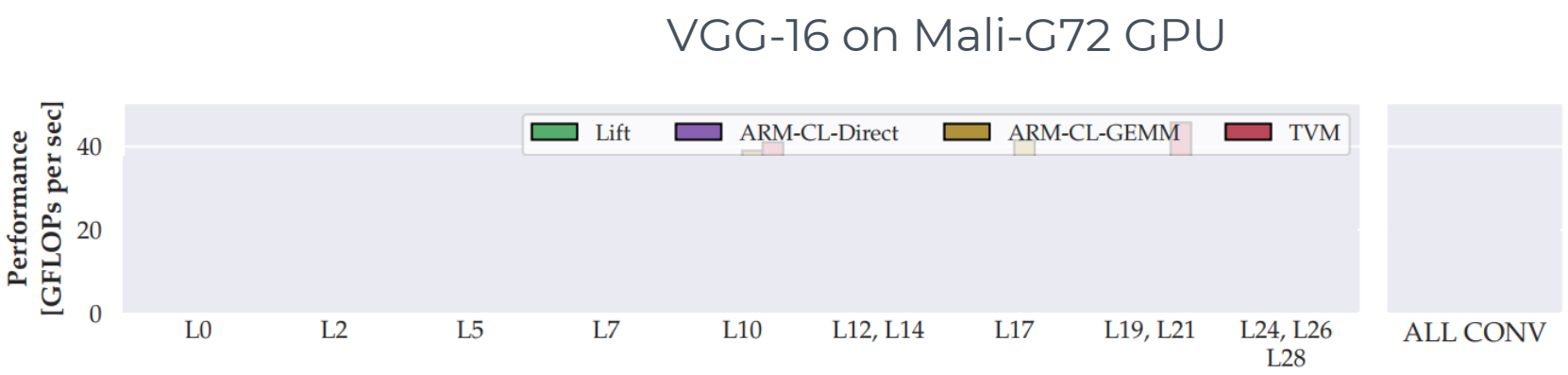
- Peak performance after 95 minutes

Results: Exploration Efficiency

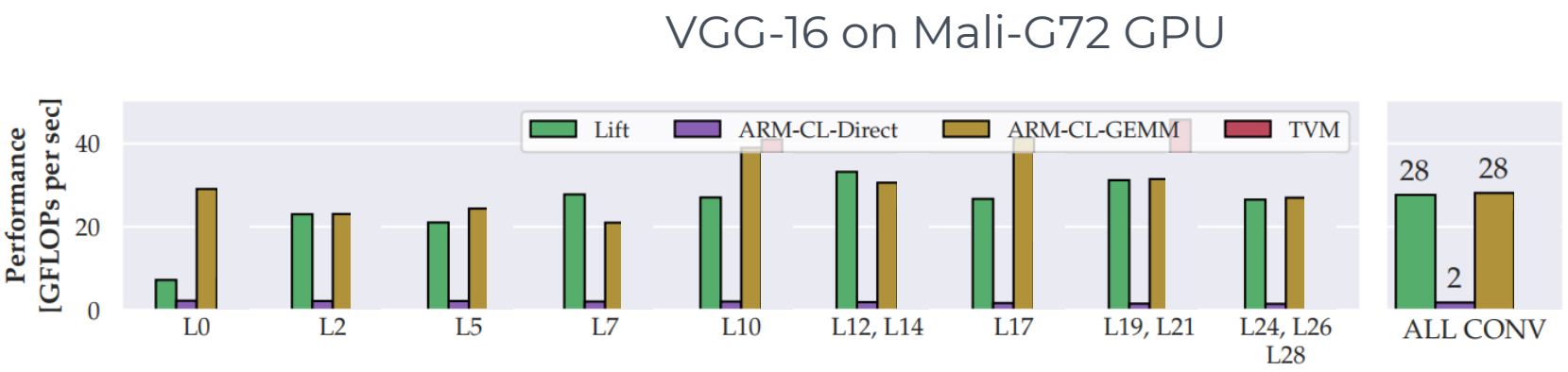


- Peak performance after 95 minutes
- Peaks before the random approach produces even 1 result (a bad one)

Results: Performance

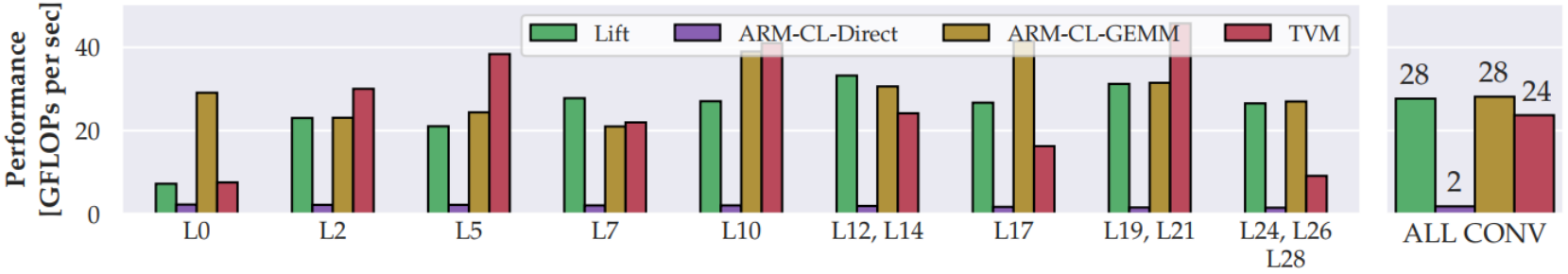


Results: Performance



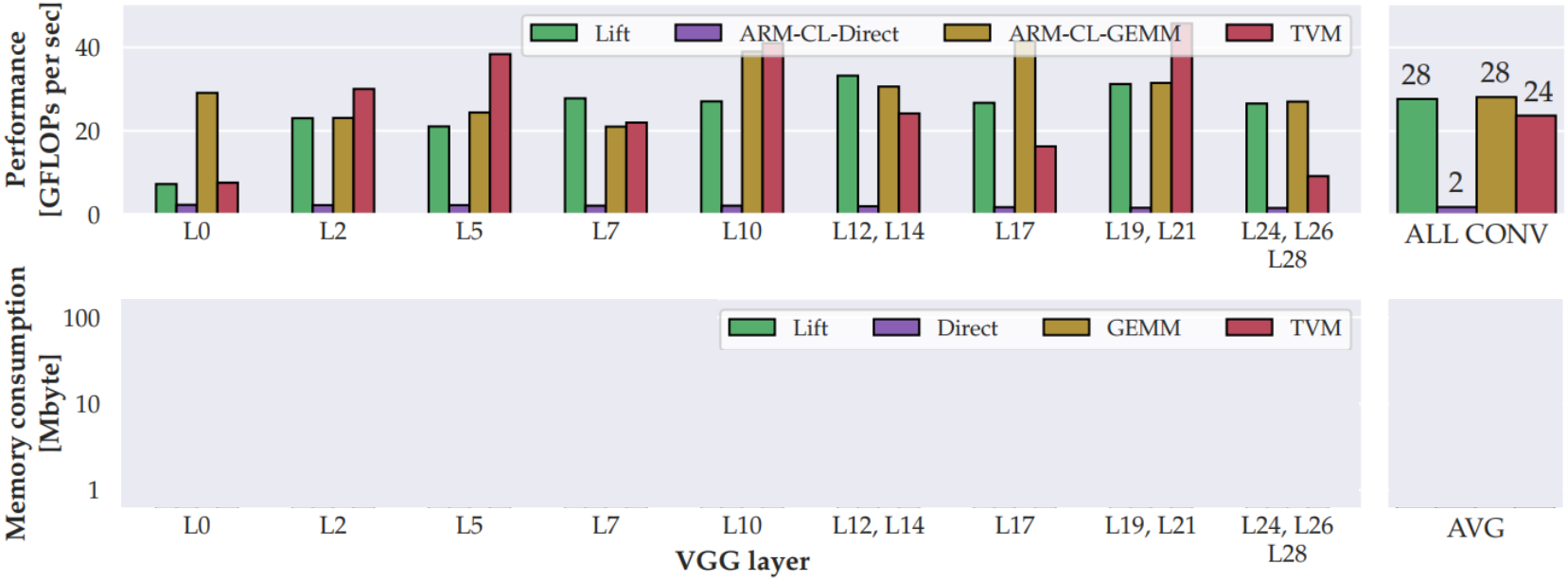
Results: Performance

VGG-16 on Mali-G72 GPU



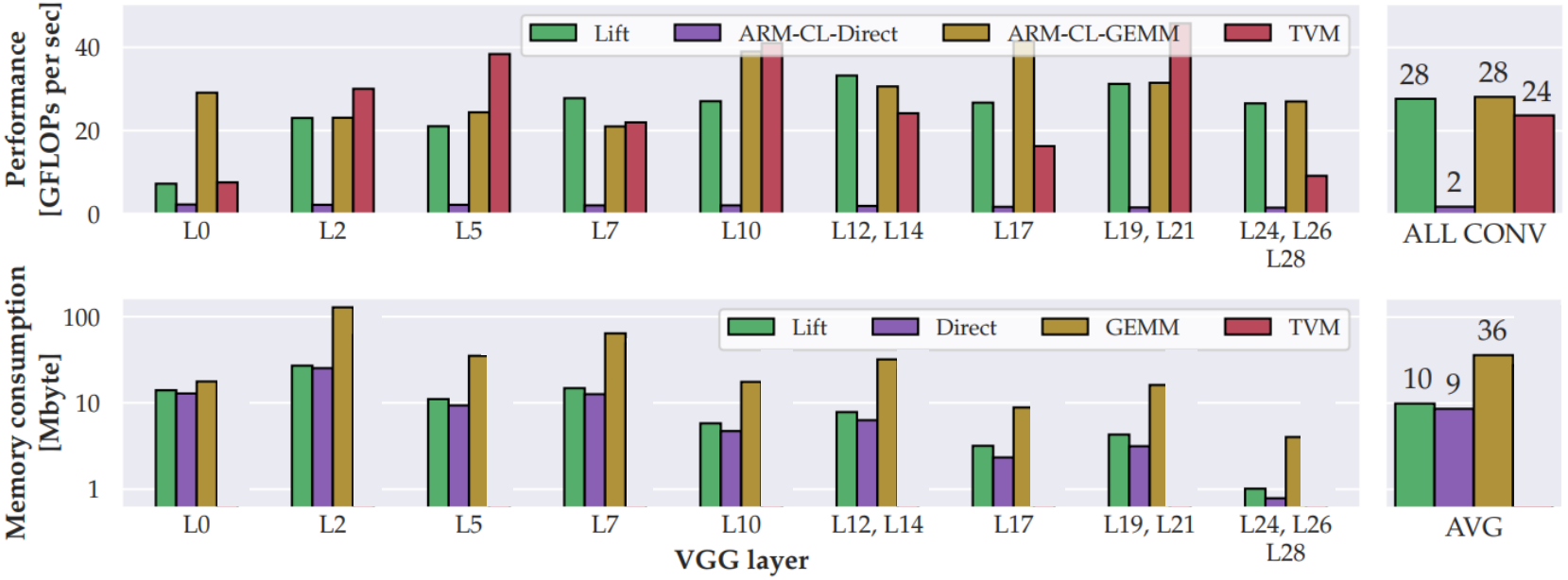
Results:
Performance & Memory

VGG-16 on Mali-G72 GPU



Results:
Performance & Memory

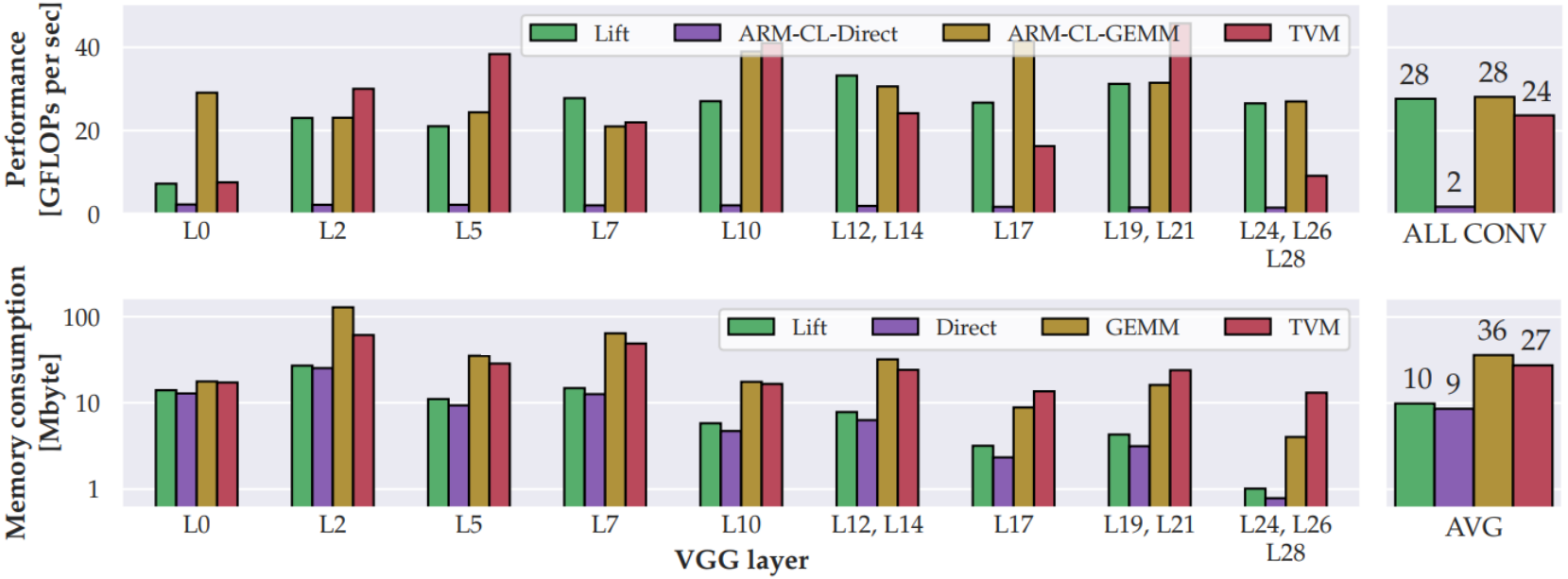
VGG-16 on Mali-G72 GPU



- **3.6x** less memory on average than ARM-CL GEMM

Results:
Performance & Memory

VGG-16 on Mali-G72 GPU



- **3.6x** less memory on average than ARM-CL GEMM
- **2.7x** less memory on average than TVM

Conclusions

Expressive IR + **Given application** + **Given set of optimizations**

Custom-tailored constraints

- Automatic parallelization is too complex to navigate using hand-coded heuristics
  - Deeply nested scheduling model on GPUs
- Constraints help prune the search space

We are Open Source!

Project: www.lift-project.org

Source: https://github.com/lift-project/lift

Artifact: https://gitlab.com/naummo/liftpar-cc-2022-artifact

Naums Mogers
University of Edinburgh, UK
naums.mogers@ed.ac.uk

Lu Li
University of Edinburgh, UK
lu.li@ed.ac.uk

Valentin Radu
University of Sheffield, UK
v.radu@sheffield.ac.uk

Christophe Dubach
McGill University, Canada
christophe.dubach@mcgill.ca