Technische Universiteit Eindhoven University of Technology

Optimization Through Recomputation in the Polyhedral Model

By Mike Jongen,
Luc Waeijen,
Roel Jordans,
Lech Jóźwiak,
Henk Corporaal.

Contents

Technische Universiteit
Eindhoven
University of Technology

Introduction

Introduction

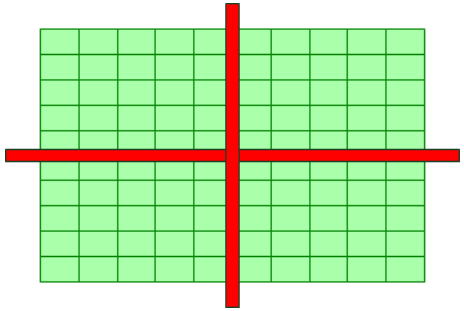
- (Mobile) systems use more artificial neural networks
  - Artificial vision
  - Image processing
  - Speech recognition

- Large amount of data accesses

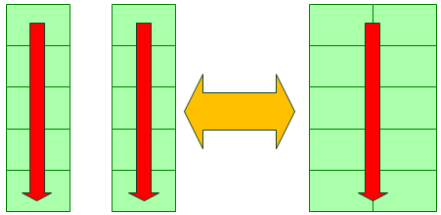
- Can be improved by code transformations

Current possibilities and extensions

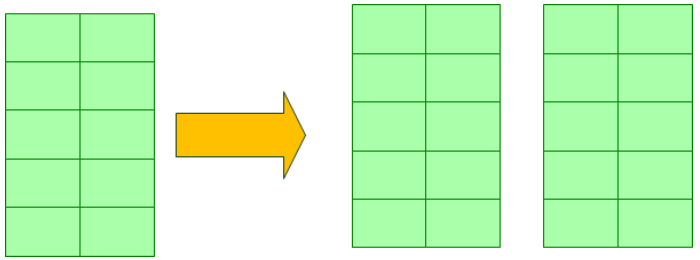
- Tiling



- Fusion
- Distribution



- Recomputation/overlapped tiling
  - Allows for better paralellism
  - Reduces memory traffic



This paper

- An example CNN application which includes recompute
- Extension of Polly
- Demonstration of the effectiveness of recomputation

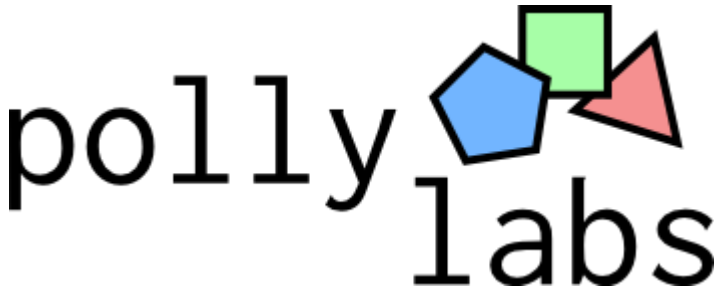
Related Work

Automated polyhedral optimization frameworks

- Greatly reduce the effort of translating the original network description into an optimized form
- Automatically verifying the validity
- Different options: Polly, R-Stream-TF, and PPCG
- None of these frameworks provides a method of including recomputation in the optimization space

Why do we use Polly

- Uses the Polyhedral model for optimizations
- Direct integration with the LLVM compiler framework
- Adjustable
  - Add extra functionality
  - User defined schedules
  - Automate the process

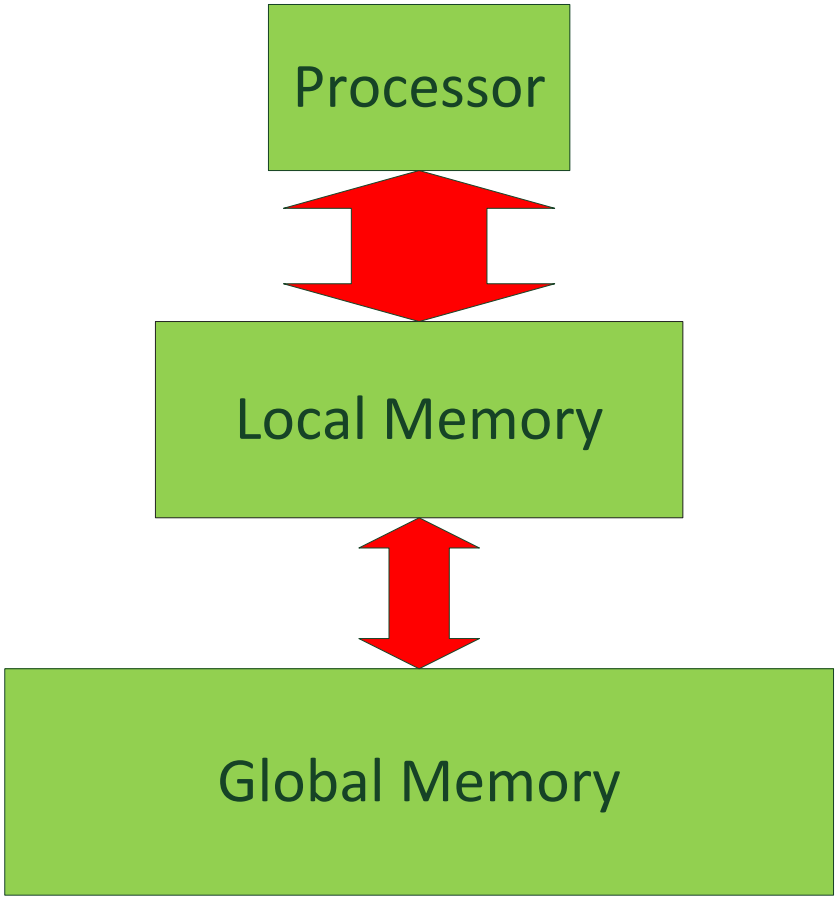


Technische Universiteit
Eindhoven
University of Technology

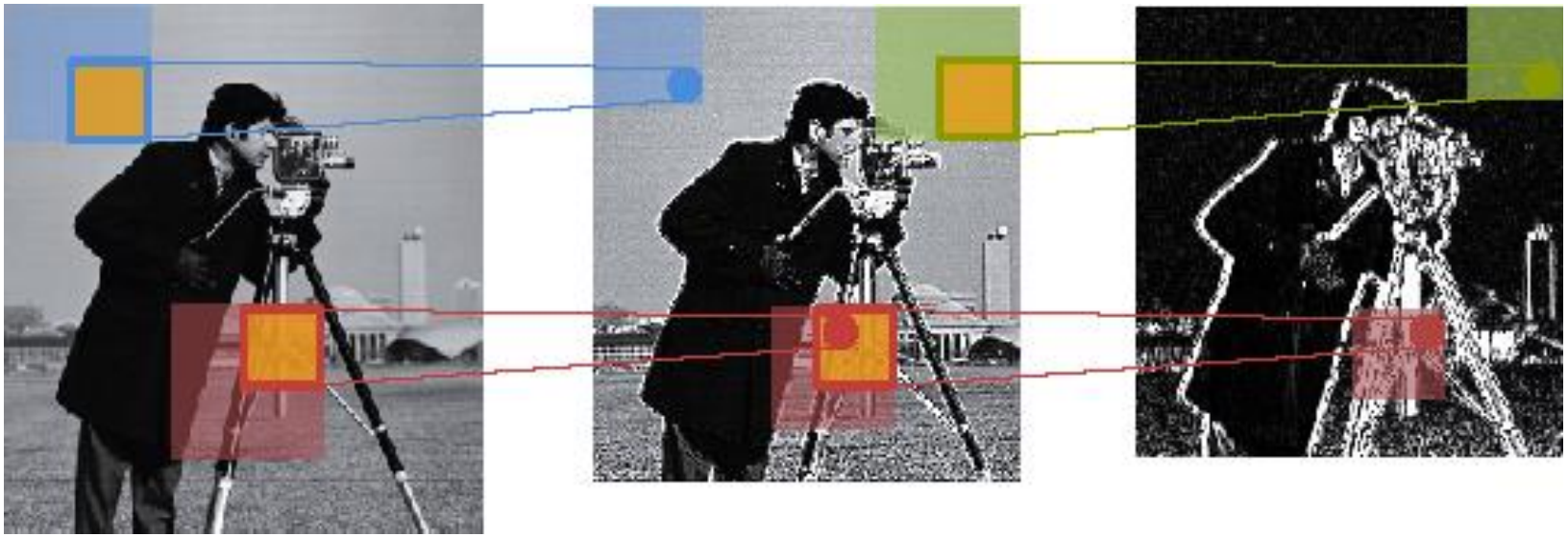
Optimizing Through Recompute

Where innovation starts

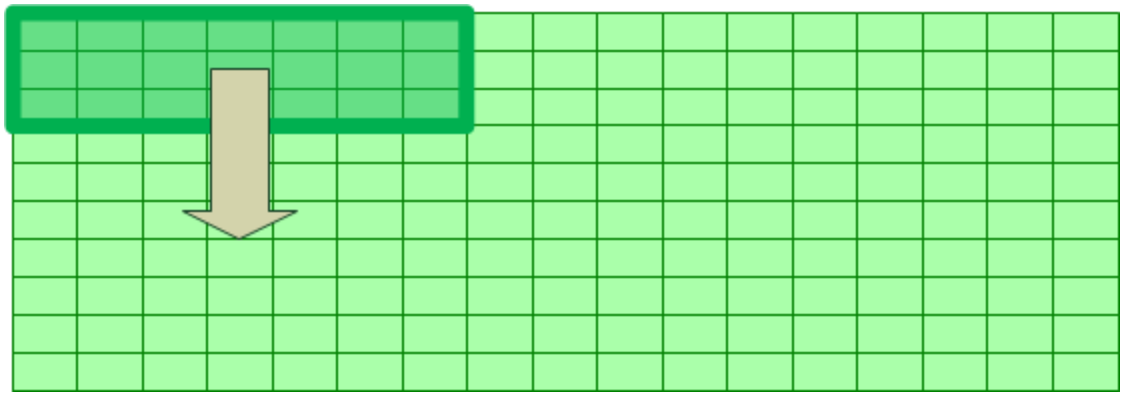
System Architecture



Educational Example

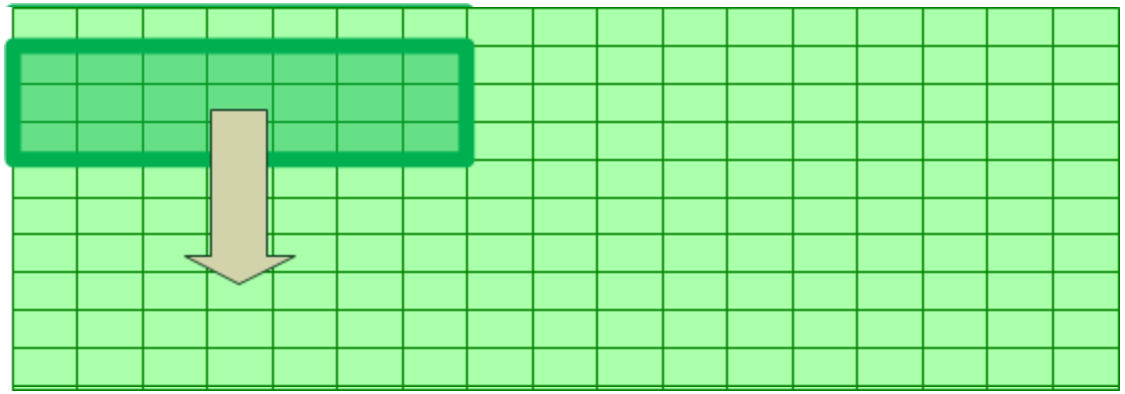


Inter Tile Reuse



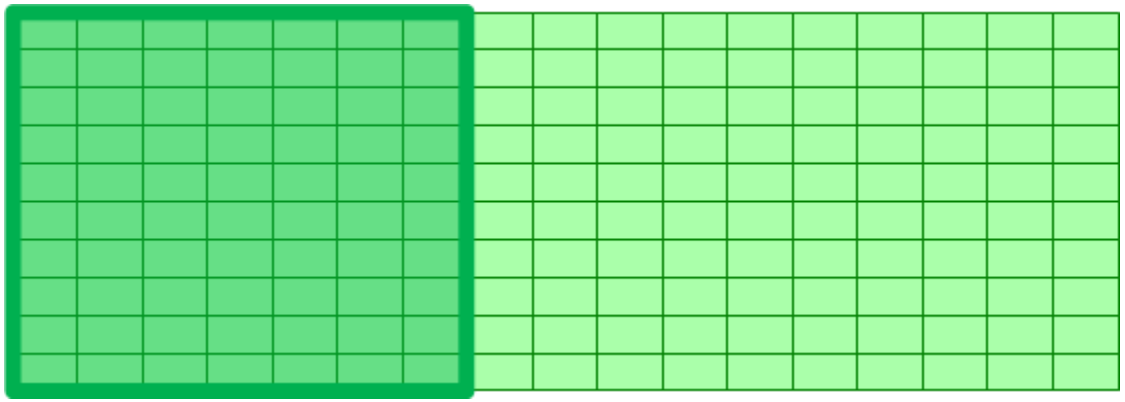
Stored Part of the intermediate image

Inter Tile Reuse

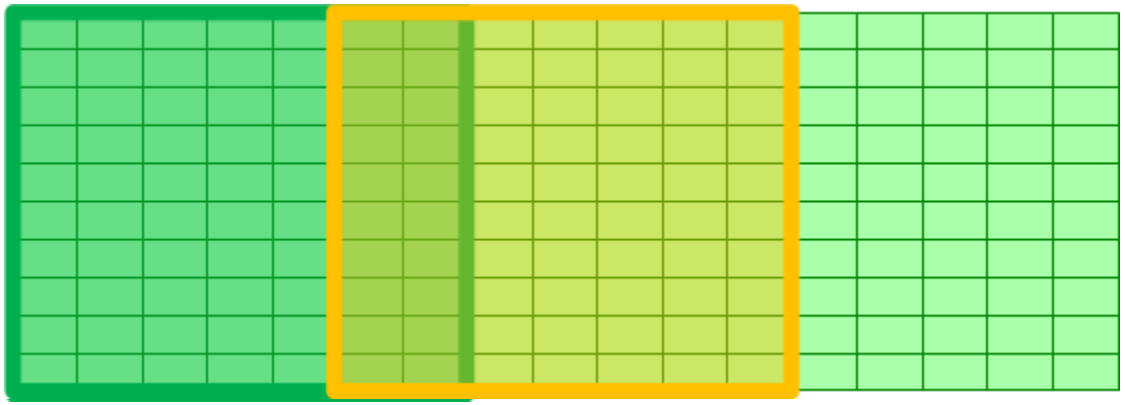


Stored Part of the intermediate image

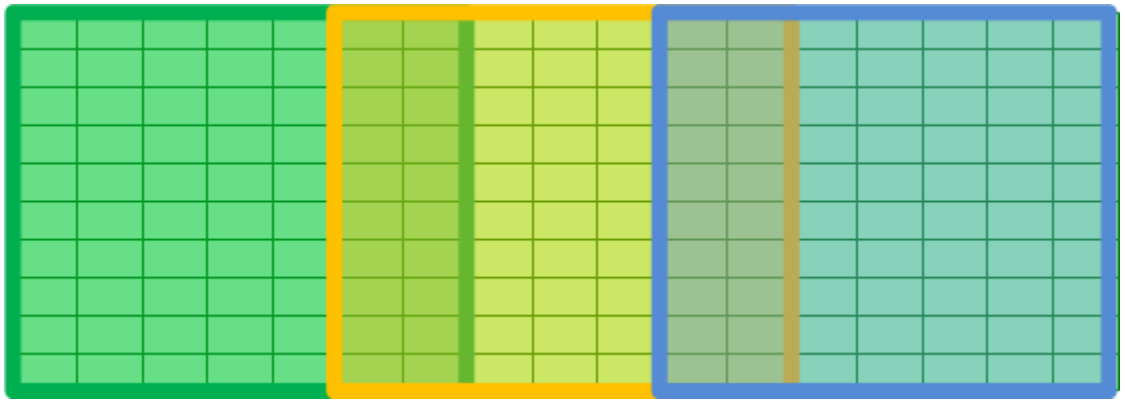
Inter Tile Reuse



Inter Tile Reuse



Other Dimensions



Methods to handle overlap

- Store the overlap globally
- Store the overlap locally
- Recompute the overlap

Global Method

- Pixels are stored externally
- Small buffer size
- Expensive memory accesses

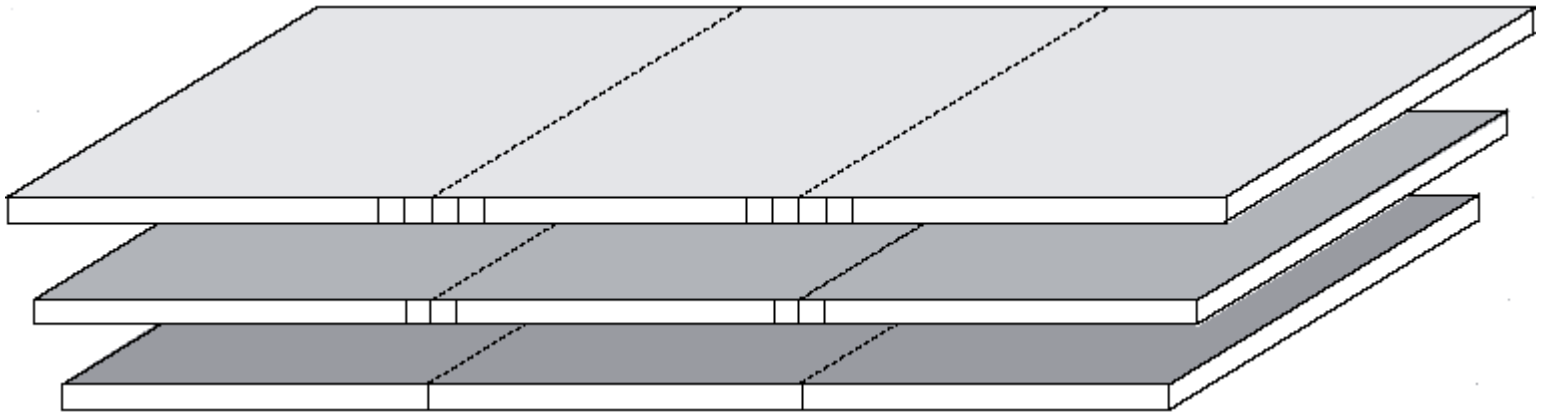
Local Method

- Pixels are stored locally
- Larger buffers required
- Cheaper accesses

Recomputation Method

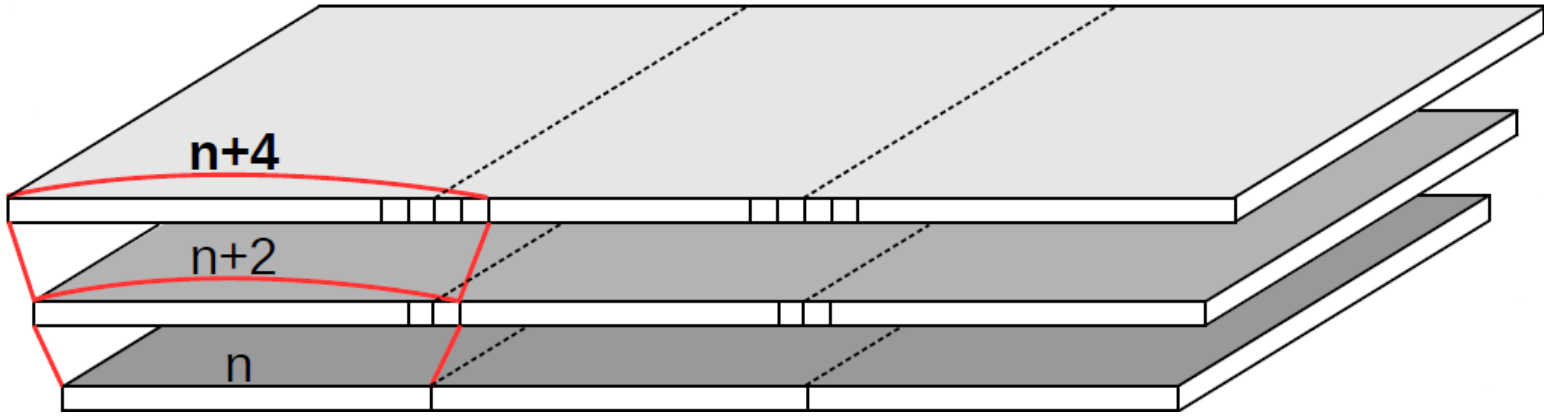
- Recomputes the pixels
- No extra memory required
- No extra accesses required
- More computations are required

Recomputation Tradeoffs



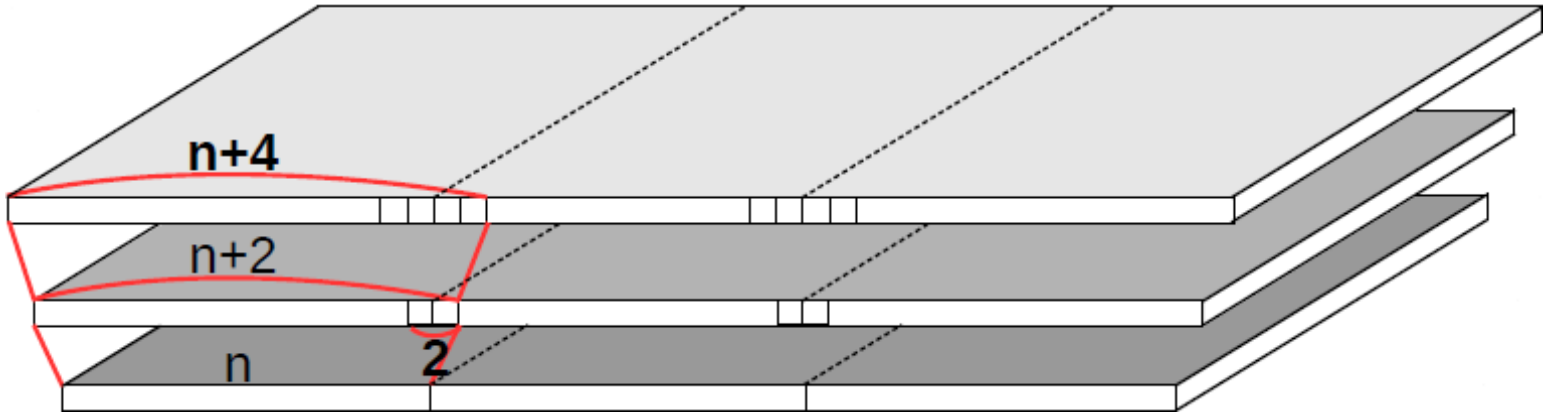
Recomputation Tradeoffs

Storing the overlap



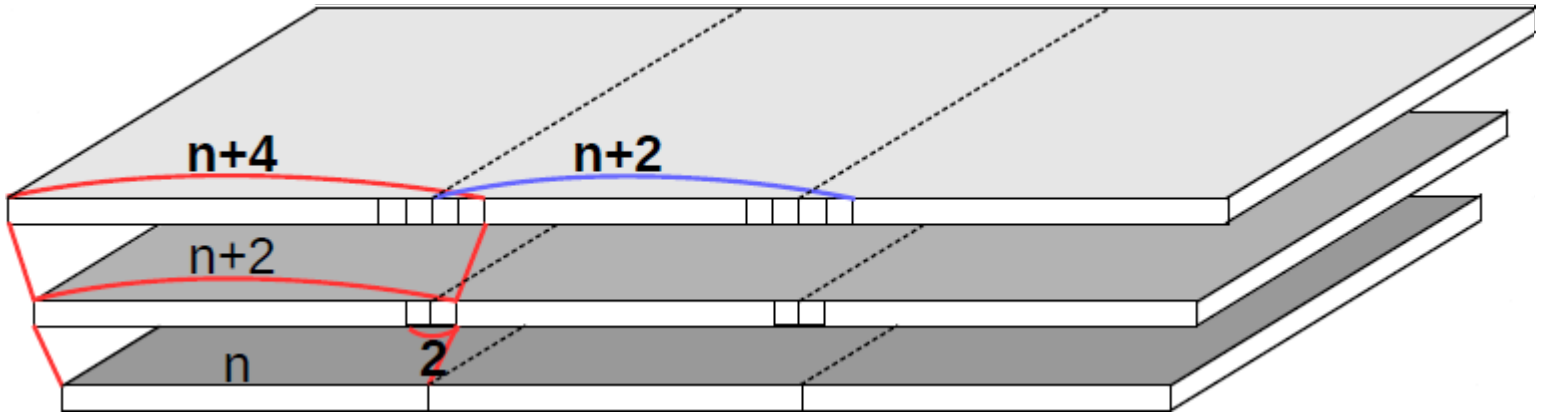
Recomputation Tradeoffs

Storing the overlap



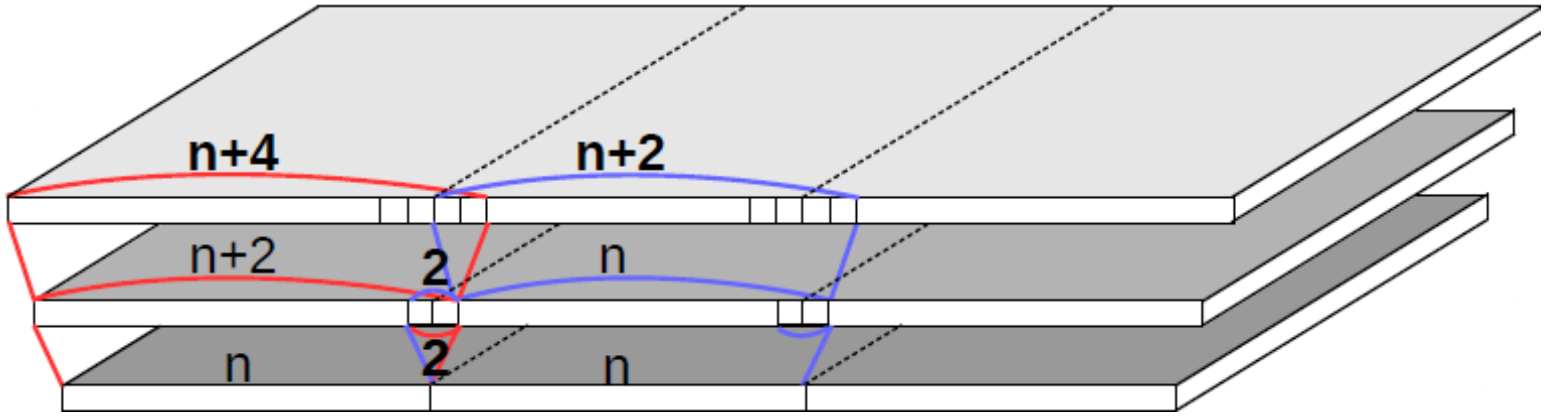
Recomputation Tradeoffs

Storing the overlap



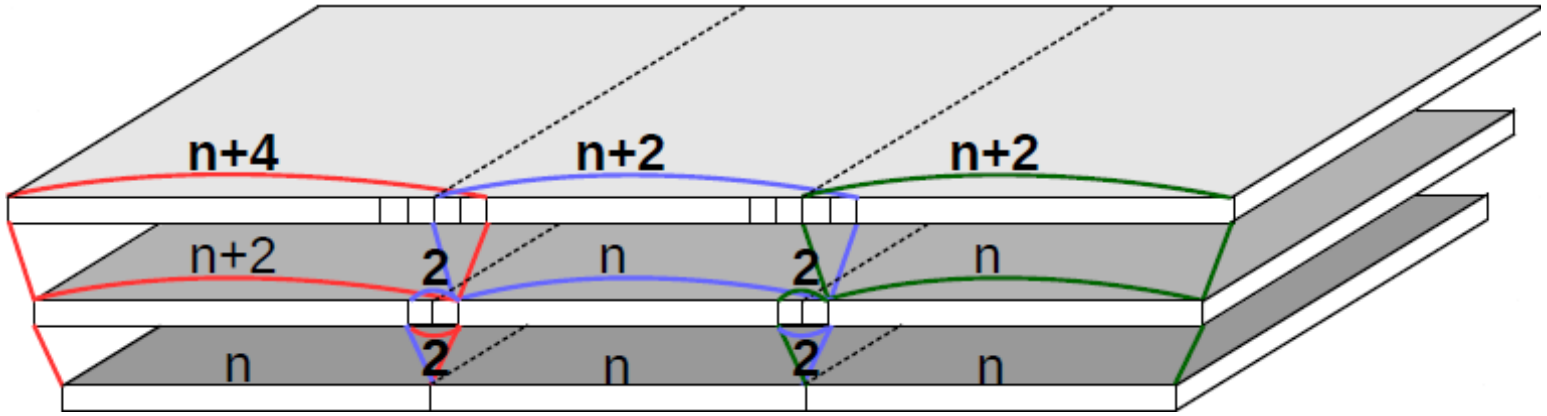
Recomputation Tradeoffs

Storing the overlap



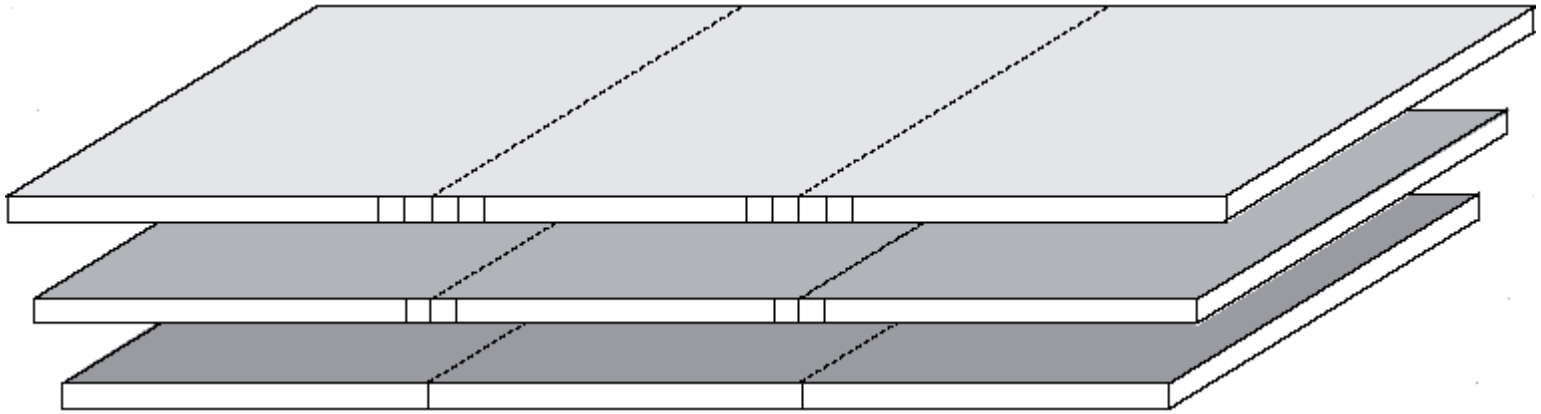
Recomputation Tradeoffs

Storing the overlap



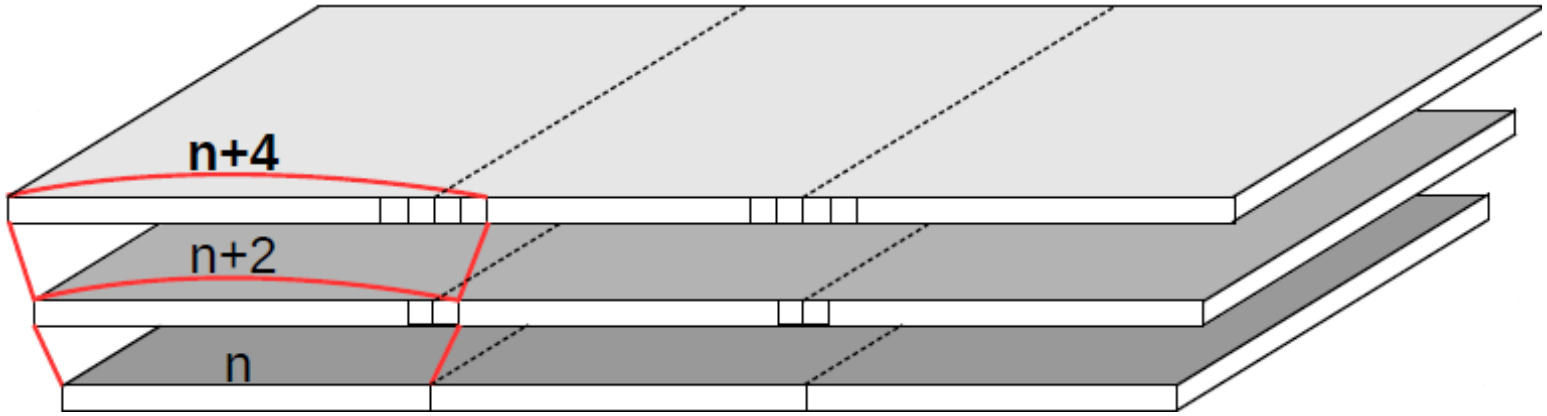
Recomputation Tradeoffs

Recomputing the overlap



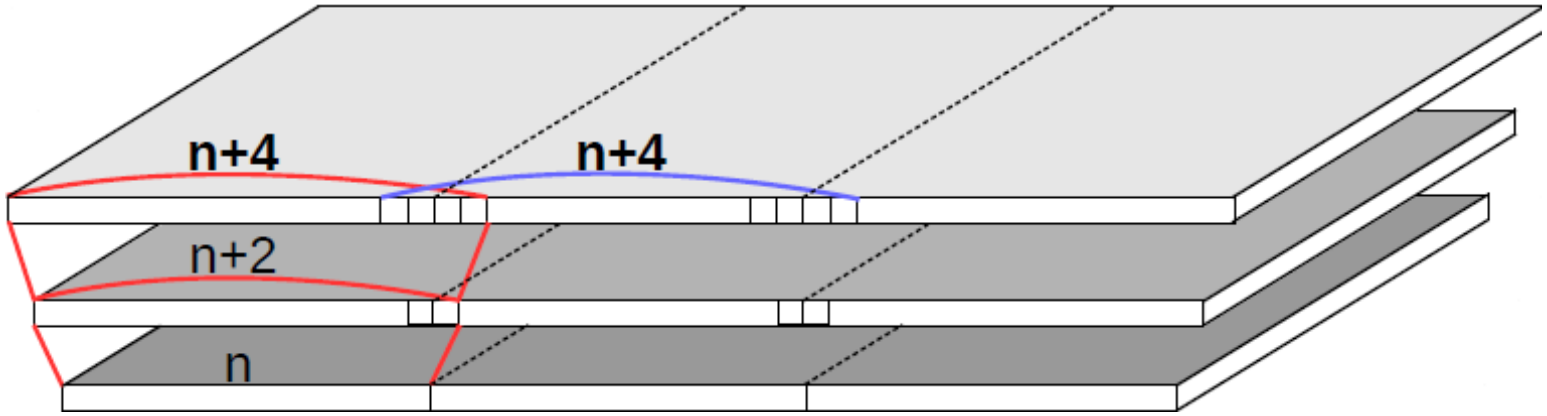
Recomputation Tradeoffs

Recomputing the overlap



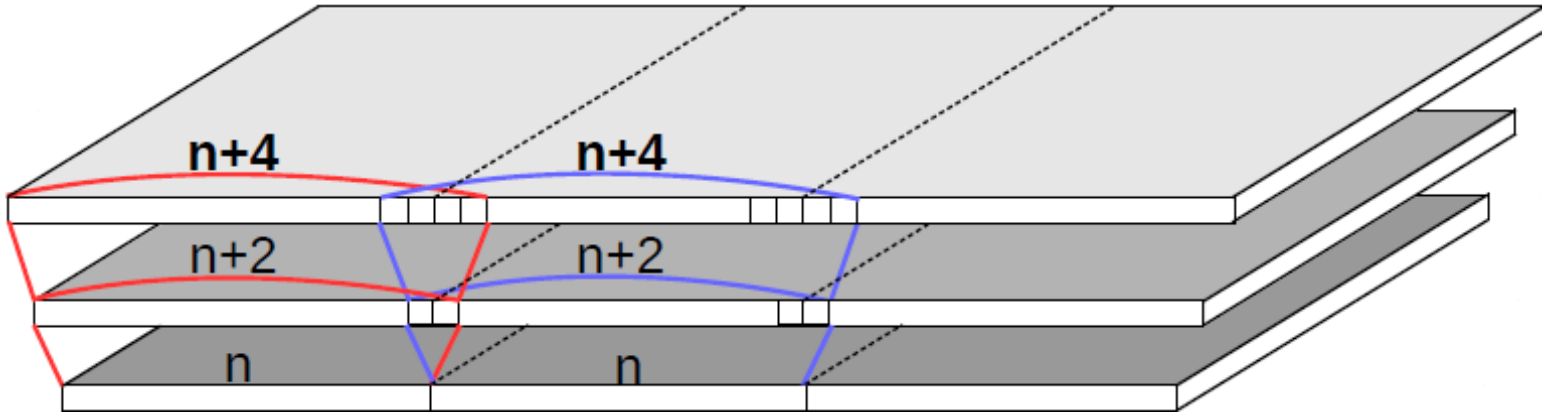
Recomputation Tradeoffs

Recomputing the overlap



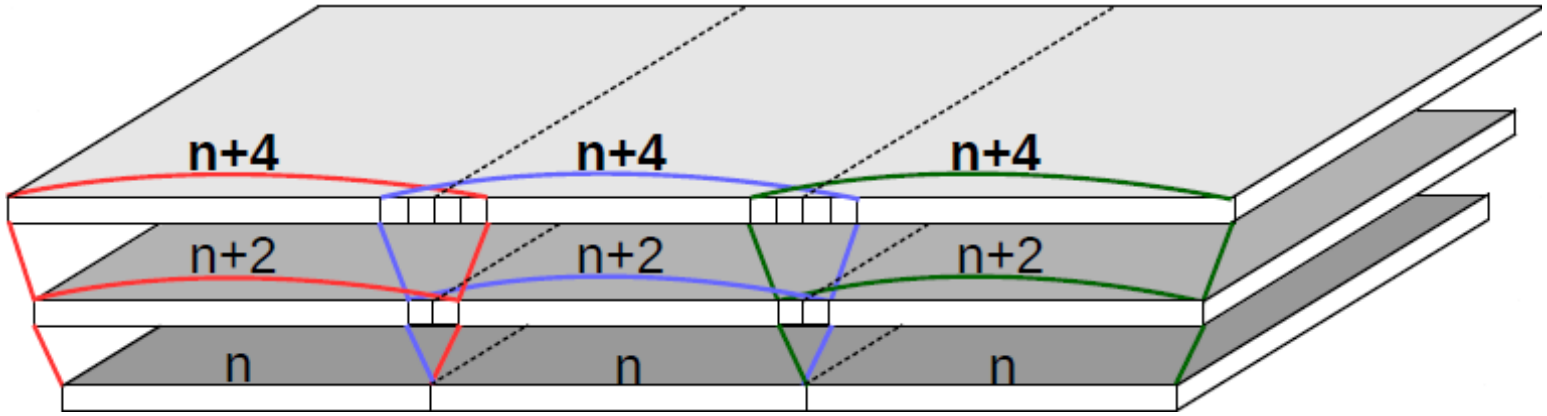
Recomputation Tradeoffs

Recomputing the overlap



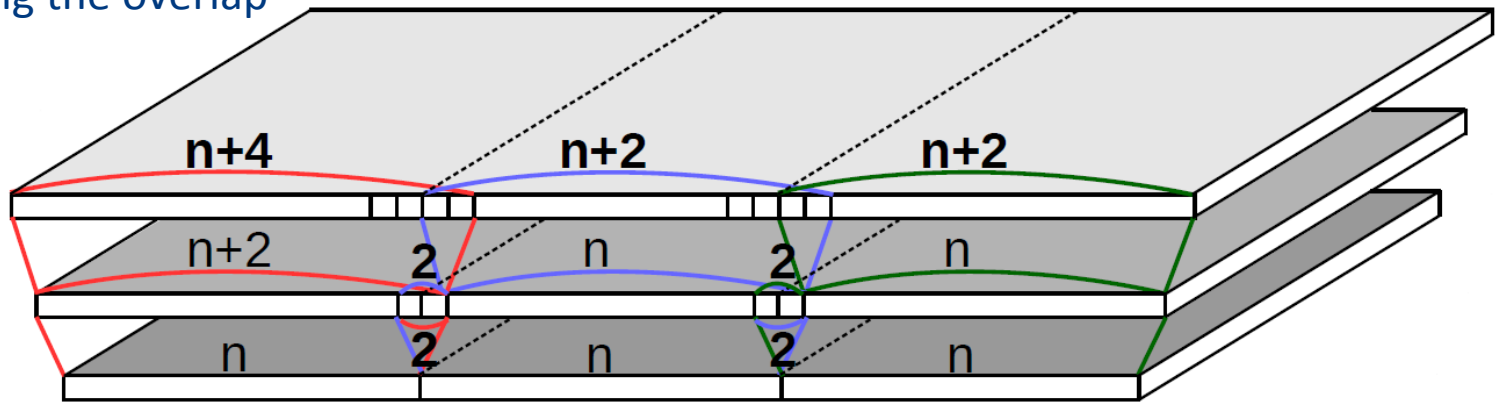
Recomputation Tradeoffs

Recomputing the overlap

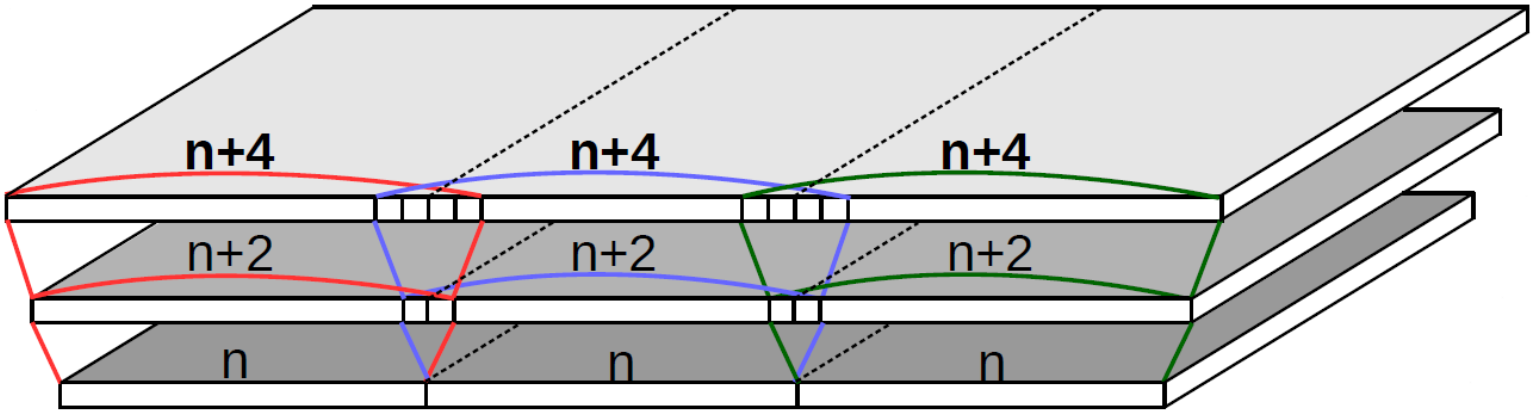


Recomputation Tradeoffs

Storing the overlap



Recomputing the overlap



Technische Universiteit
Eindhoven
University of Technology

Polyhedral Modeling

The Polyhedral Model and Recomputation

- Execution order is defined by the schedule

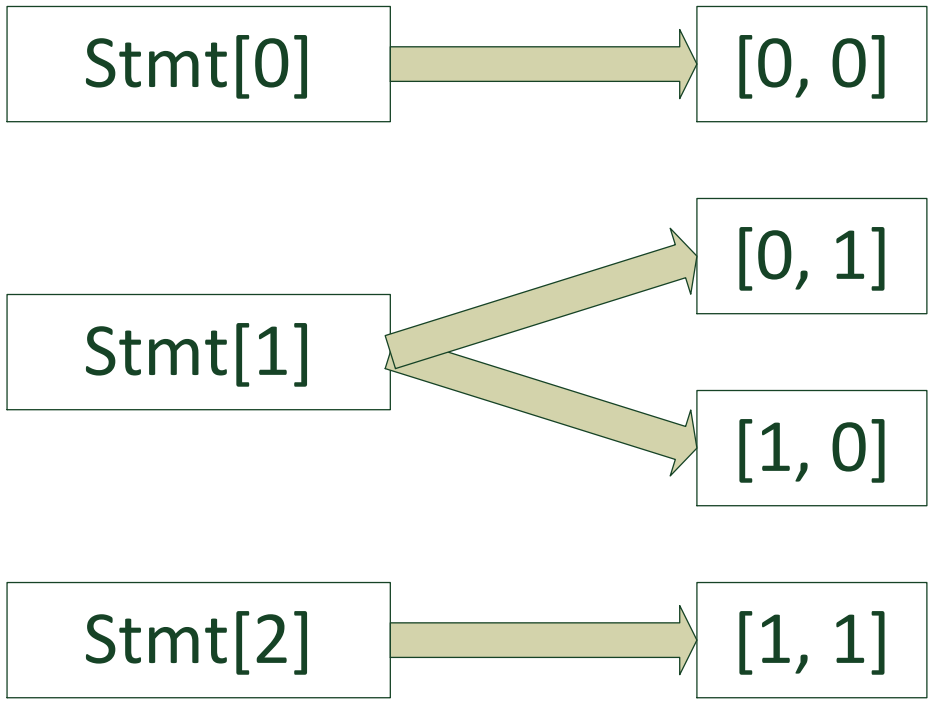
- Schedule is singular valued
  - One execution time per statement
  - One statement per execution time

- Recomputation:
  - Statements are executed multiple times
  - Non-singular valued schedules are required

Including Recomputation

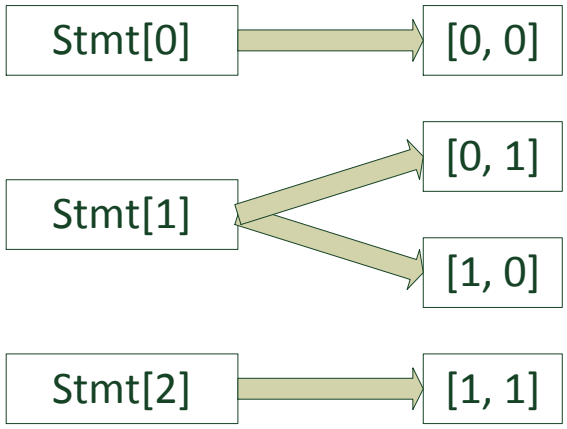
- Support for non-singular valued schedules
- Transforming non-singular valued schedules to singular valued schedules

Example



Example

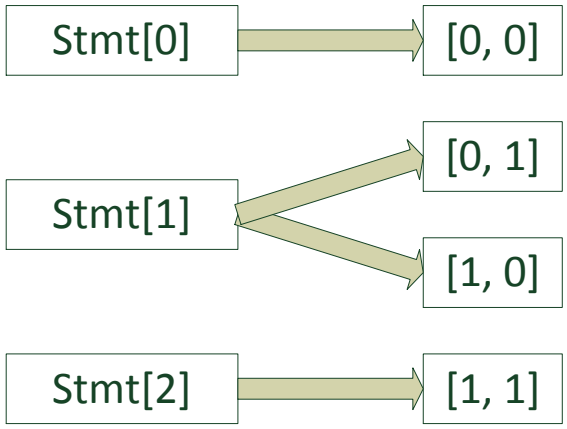
Old Schedule



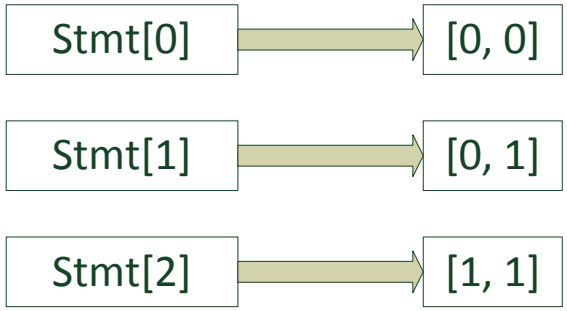
Data: OriginalSchedule
Result: NewSchedule
set Lexmin to the lexographical minimum of OriginalSchedule;
if *OriginalSchedule is equal to Lexmin* **then**
    set newSchedule to OriginalSchedule;
else
    add a dimension to newSchedule;
    set i to 0;
    RestofSchedule = OriginalSchedule - Lexmin;
    while *RestofSchedule is not empty* **do**
        add Lexmin to newSchedule with the new dimension set to i;
        i++;
        set Lexmin to the lexographical minimum of RestOfSchedule;
        RestofSchedule = RestofSchedule- Lexmin;
    end
    add Lexmin to newSchedule with the new dimension set to i;
end

Example

Old Schedule



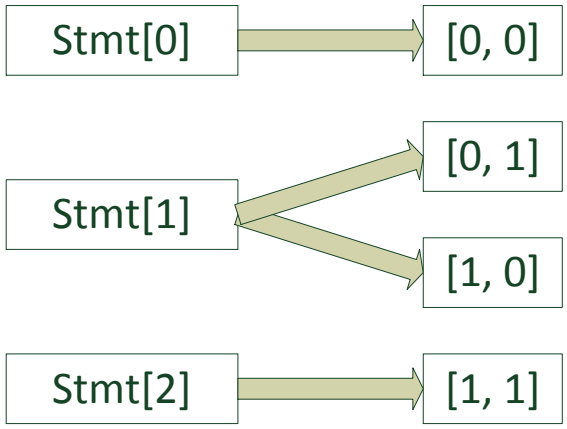
Lexicographical Minimum



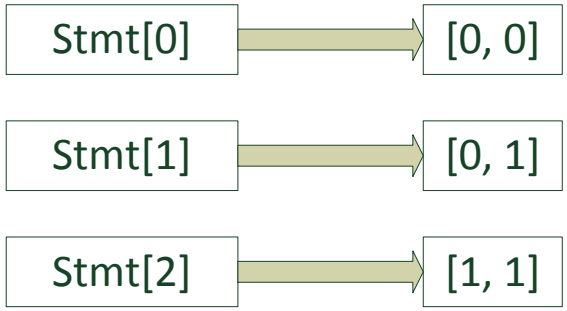
Data: OriginalSchedule
Result: NewSchedule
set Lexmin to the lexographical minimum of OriginalSchedule;
if *OriginalSchedule is equal to Lexmin* **then**
    set newSchedule to OriginalSchedule;
else
    add a dimension to newSchedule;
    set i to 0;
    RestofSchedule = OriginalSchedule - Lexmin;
    while *RestofSchedule is not empty* **do**
        add Lexmin to newSchedule with the new dimension set to i;
        i++;
        set Lexmin to the lexographical minimum of RestOfSchedule;
        RestofSchedule = RestofSchedule- Lexmin;
    end
    add Lexmin to newSchedule with the new dimension set to i;
end

Example

Old Schedule



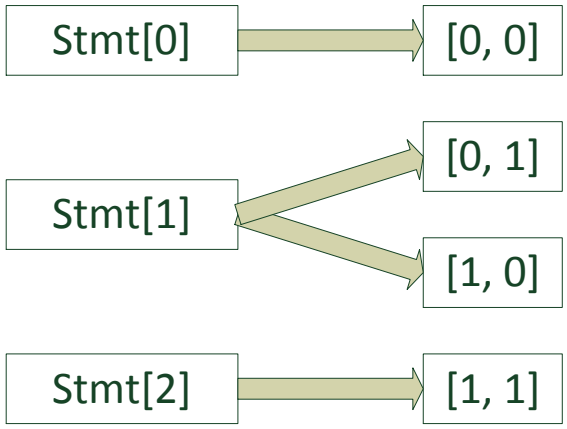
Lexicographical Minimum



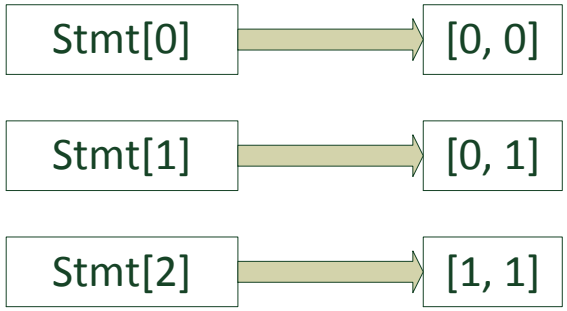

```
Data: OriginalSchedule
Result: NewSchedule
set Lexmin to the lexographical minimum of
OriginalSchedule;
if OriginalSchedule is equal to Lexmin then
    set newSchedule to OriginalSchedule;
else
    add a dimension to newSchedule;
    set i to 0;
    RestofSchedule = OriginalSchedule - Lexmin;
    while RestofSchedule is not empty do
        add Lexmin to newSchedule with the new
        dimension set to i;
        i++;
        set Lexmin to the lexographical minimum of
        RestOfSchedule;
        RestofSchedule = RestofSchedule- Lexmin;
    end
    add Lexmin to newSchedule with the new
    dimension set to i;
end
```

Example

Old Schedule



Lexicographical Minimum

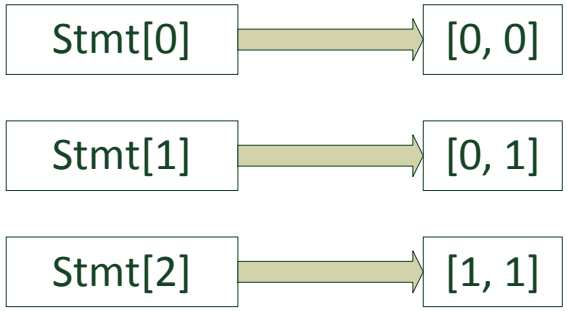


Data: OriginalSchedule
Result: NewSchedule
set Lexmin to the lexographical minimum of OriginalSchedule;
if *OriginalSchedule is equal to Lexmin* **then**
    set newSchedule to OriginalSchedule;
else
    add a dimension to newSchedule;
    set i to 0;
    RestofSchedule = OriginalSchedule - Lexmin;
    while *RestofSchedule is not empty* **do**
        add Lexmin to newSchedule with the new dimension set to i;
        i++;
        set Lexmin to the lexographical minimum of RestOfSchedule;
        RestofSchedule = RestofSchedule- Lexmin;
    end
    add Lexmin to newSchedule with the new dimension set to i;
end

Example

Rest of Schedule

Lexicographical Minimum

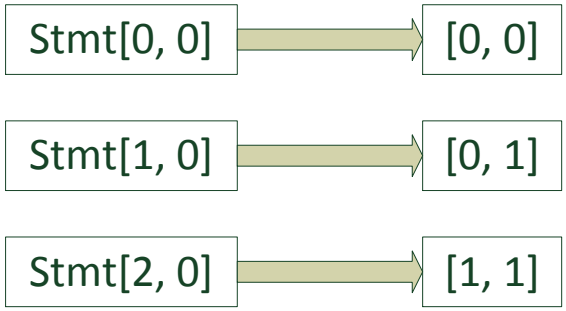



```
Data: OriginalSchedule
Result: NewSchedule
set Lexmin to the lexographical minimum of
OriginalSchedule;
if OriginalSchedule is equal to Lexmin then
    set newSchedule to OriginalSchedule;
else
    add a dimension to newSchedule;
    set i to 0;
    RestofSchedule = OriginalSchedule - Lexmin;
    while RestofSchedule is not empty do
        add Lexmin to newSchedule with the new
        dimension set to i;
        i++;
        set Lexmin to the lexographical minimum of
        RestOfSchedule;
        RestofSchedule = RestofSchedule- Lexmin;
    end
    add Lexmin to newSchedule with the new
    dimension set to i;
end
```

Example

Rest of Schedule

New Schedule

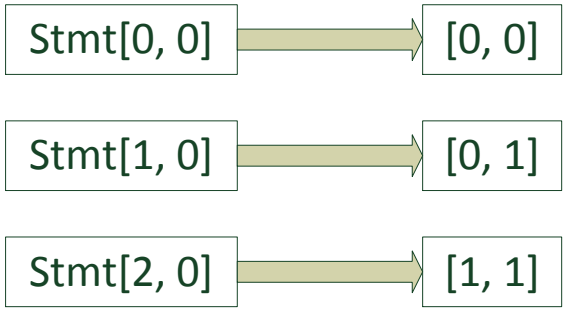



```
Data: OriginalSchedule
Result: NewSchedule
set Lexmin to the lexographical minimum of
OriginalSchedule;
if OriginalSchedule is equal to Lexmin then
    set newSchedule to OriginalSchedule;
else
    add a dimension to newSchedule;
    set i to 0;
    RestofSchedule = OriginalSchedule - Lexmin;
    while RestofSchedule is not empty do
        add Lexmin to newSchedule with the new
        dimension set to i;
        i++;
        set Lexmin to the lexographical minimum of
        RestOfSchedule;
        RestofSchedule = RestofSchedule- Lexmin;
    end
    add Lexmin to newSchedule with the new
    dimension set to i;
end
```

Example

Lexicographical Minimum

New Schedule

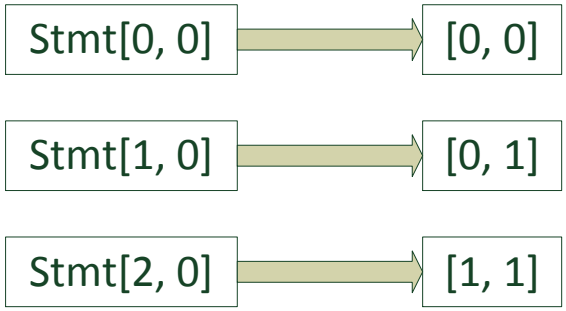



```
Data: OriginalSchedule
Result: NewSchedule
set Lexmin to the lexographical minimum of
OriginalSchedule;
if OriginalSchedule is equal to Lexmin then
    set newSchedule to OriginalSchedule;
else
    add a dimension to newSchedule;
    set i to 0;
    RestofSchedule = OriginalSchedule - Lexmin;
    while RestofSchedule is not empty do
        add Lexmin to newSchedule with the new
        dimension set to i;
        i++;
        set Lexmin to the lexographical minimum of
        RestOfSchedule;
        RestofSchedule = RestofSchedule- Lexmin;
    end
    add Lexmin to newSchedule with the new
    dimension set to i;
end
```

Example

Lexicographical Minimum

New Schedule

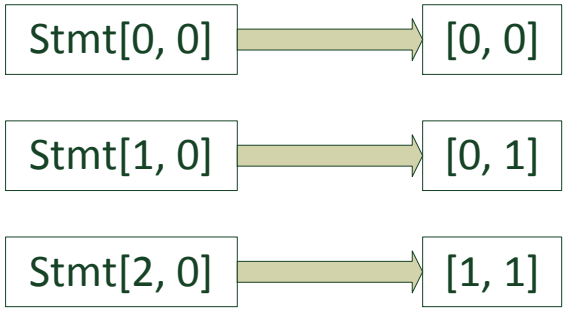



```
Data: OriginalSchedule
Result: NewSchedule
set Lexmin to the lexographical minimum of
OriginalSchedule;
if OriginalSchedule is equal to Lexmin then
    set newSchedule to OriginalSchedule;
else
    add a dimension to newSchedule;
    set i to 0;
    RestofSchedule = OriginalSchedule - Lexmin;
    while RestofSchedule is not empty do
        add Lexmin to newSchedule with the new
        dimension set to i;
        i++;
        set Lexmin to the lexographical minimum of
        RestOfSchedule;
        RestofSchedule = RestofSchedule- Lexmin;
    end
    add Lexmin to newSchedule with the new
    dimension set to i;
end
```

Example

Lexicographical Minimum

New Schedule

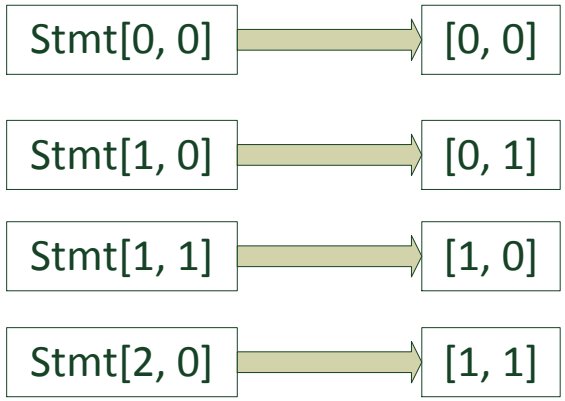



```
Data: OriginalSchedule
Result: NewSchedule
set Lexmin to the lexographical minimum of
OriginalSchedule;
if OriginalSchedule is equal to Lexmin then
    set newSchedule to OriginalSchedule;
else
    add a dimension to newSchedule;
    set i to 0;
    RestofSchedule = OriginalSchedule - Lexmin;
    while RestofSchedule is not empty do
        add Lexmin to newSchedule with the new
        dimension set to i;
        i++;
        set Lexmin to the lexographical minimum of
        RestOfSchedule;
        RestofSchedule = RestofSchedule- Lexmin;
    end
    add Lexmin to newSchedule with the new
    dimension set to i;
end
```

Example

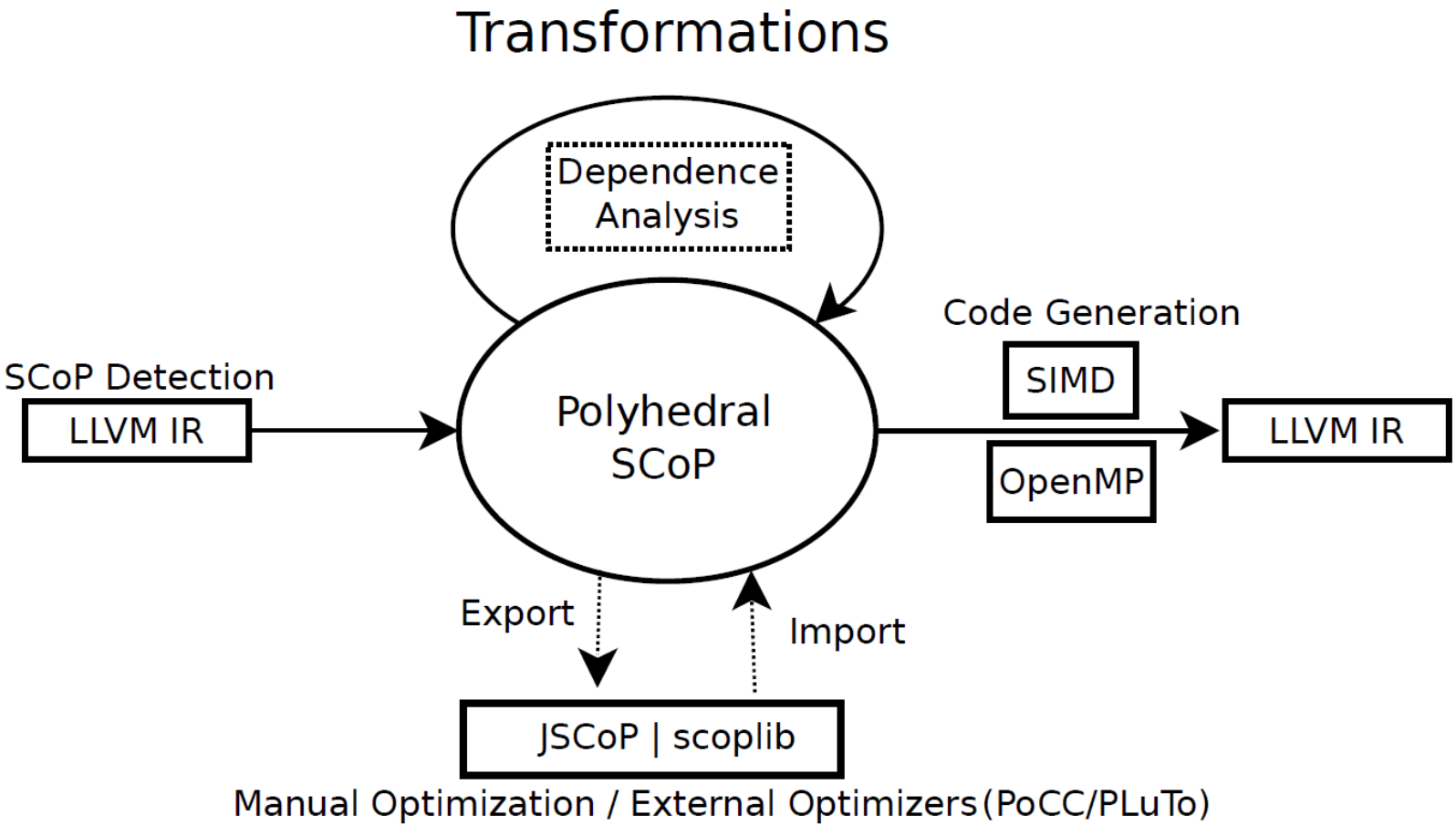
Lexicographical Minimum

New Schedule

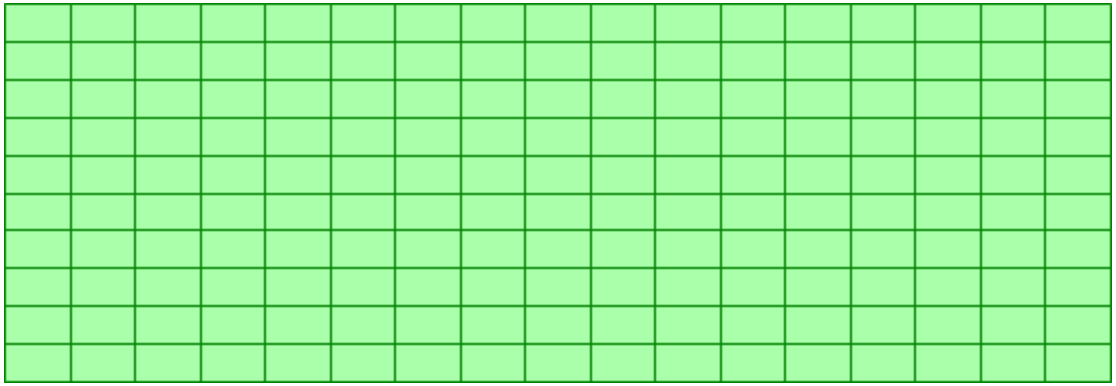



```
Data: OriginalSchedule
Result: NewSchedule
set Lexmin to the lexographical minimum of
OriginalSchedule;
if OriginalSchedule is equal to Lexmin then
    set newSchedule to OriginalSchedule;
else
    add a dimension to newSchedule;
    set i to 0;
    RestofSchedule = OriginalSchedule - Lexmin;
    while RestofSchedule is not empty do
        add Lexmin to newSchedule with the new
        dimension set to i;
        i++;
        set Lexmin to the lexographical minimum of
        RestOfSchedule;
        RestofSchedule = RestofSchedule- Lexmin;
    end
    add Lexmin to newSchedule with the new
    dimension set to i;
end
```

Including Recomputation: location

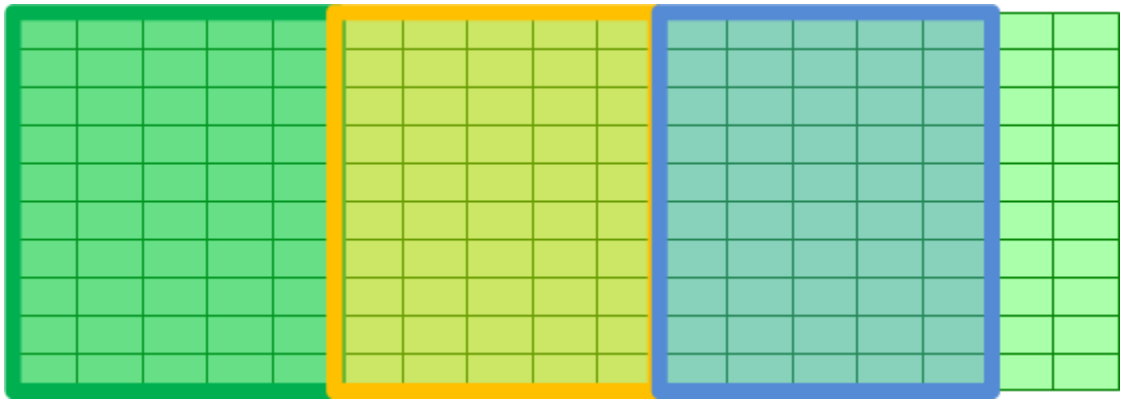


Jscop Implementation



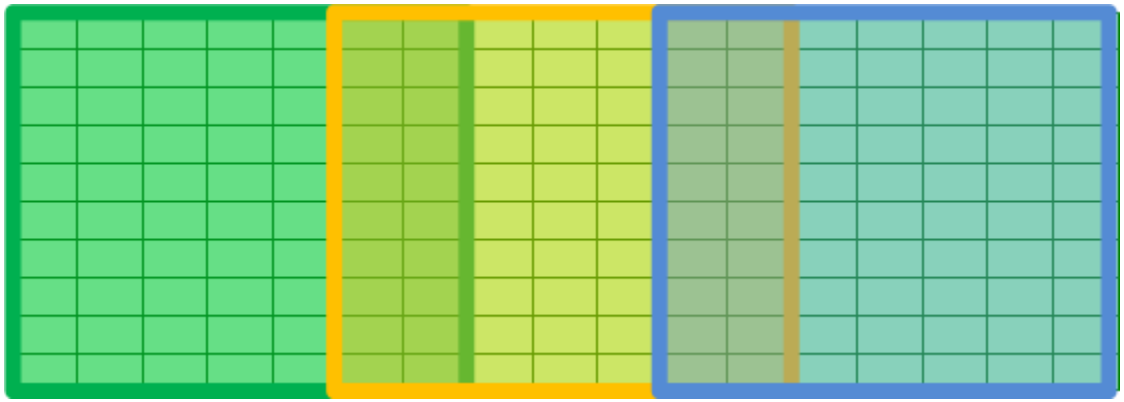
Conv[i0,i1,i2,i3] → [i0,i1,i2,i3]

Jscop Implementation



Conv[i0,i1,i2,i3] →[t0,i1,t1,i2,i3] :
0 <= t0 < no_tiles and
0 <= t1 < tilesize and
i0 = tilesize ∗ t0 + t1

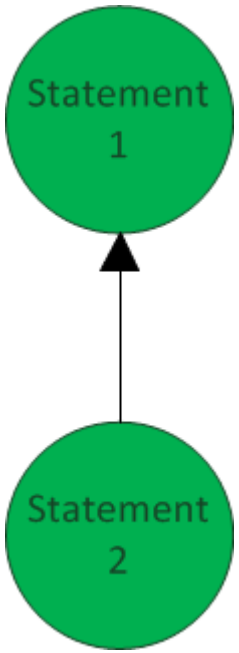
Jscop Implementation



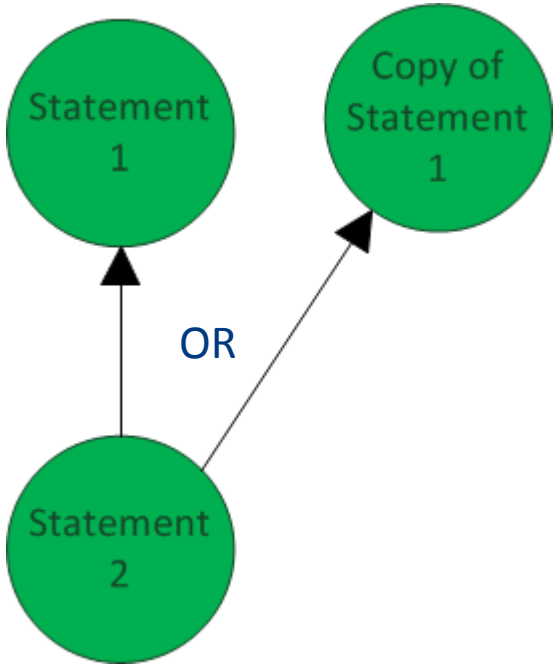
Conv[i0,i1,i2,i3] →[t0,i1,t1,i2,i3] :
0 <= t0 < no_tiles and
0 <= t1 < tilesize + overlap and
i0 = tilesize ∗ t0 + t1

Dependencies

Before



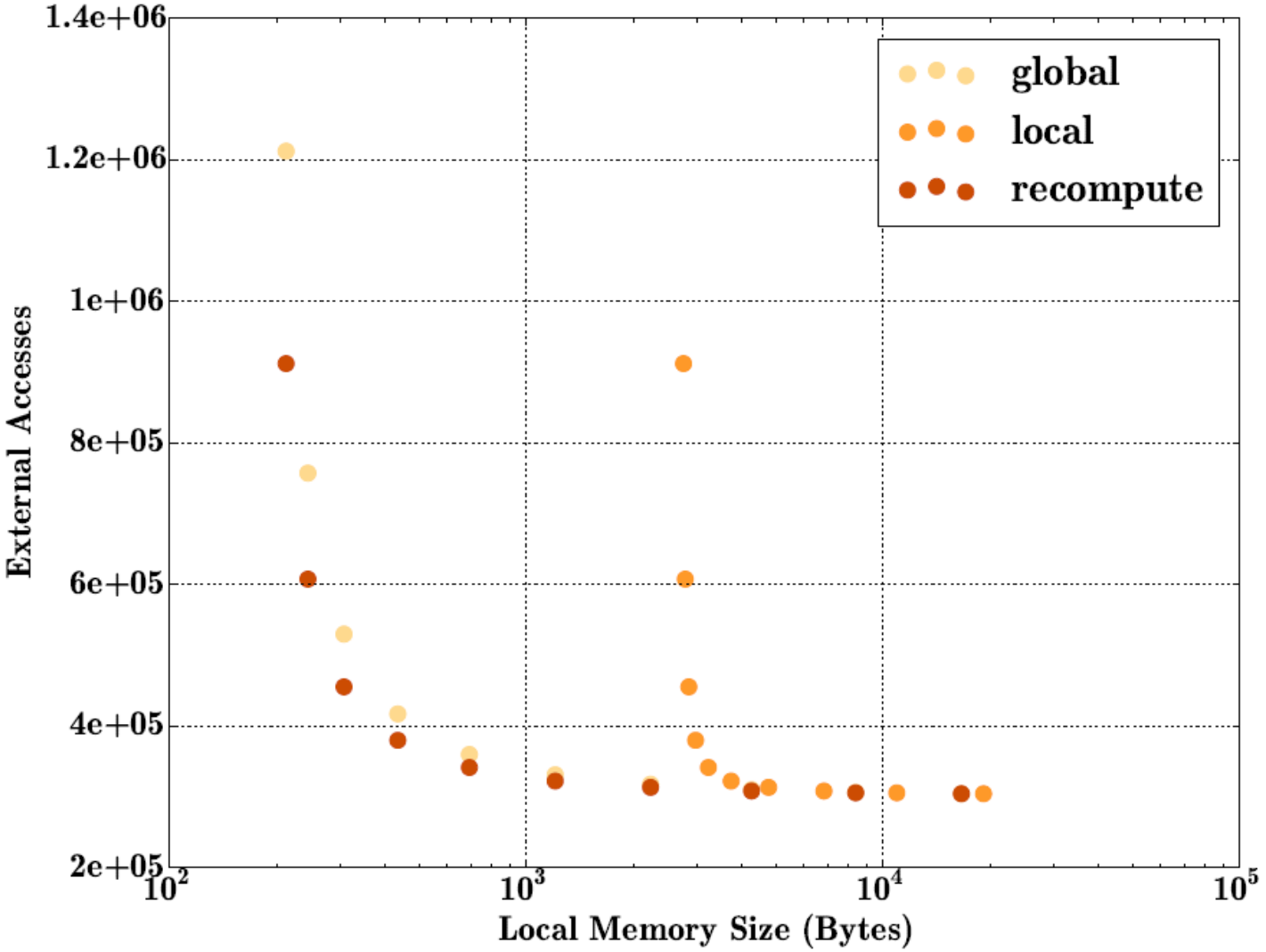
After



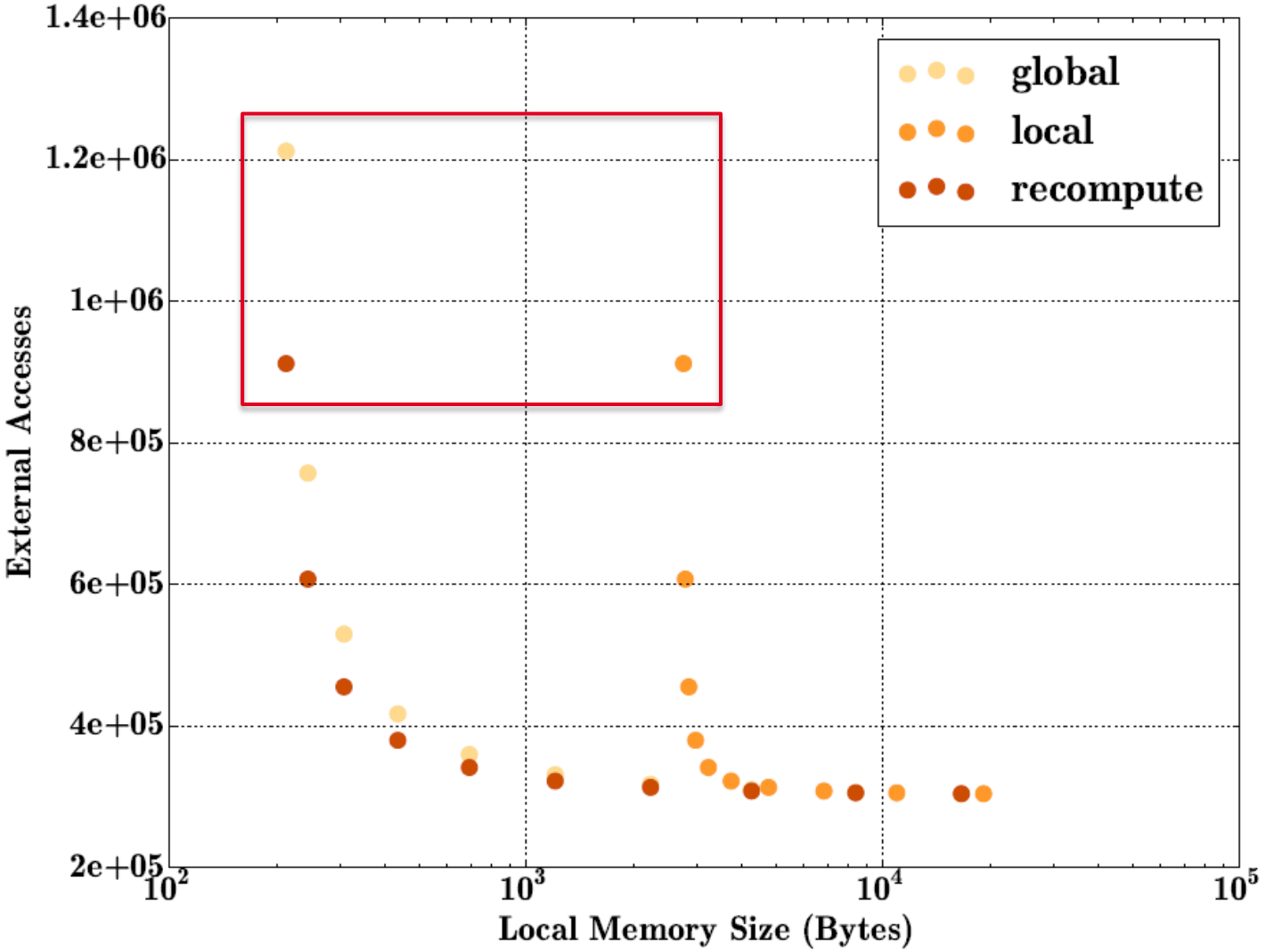
Technische Universiteit
Eindhoven
University of Technology

Experimental Results

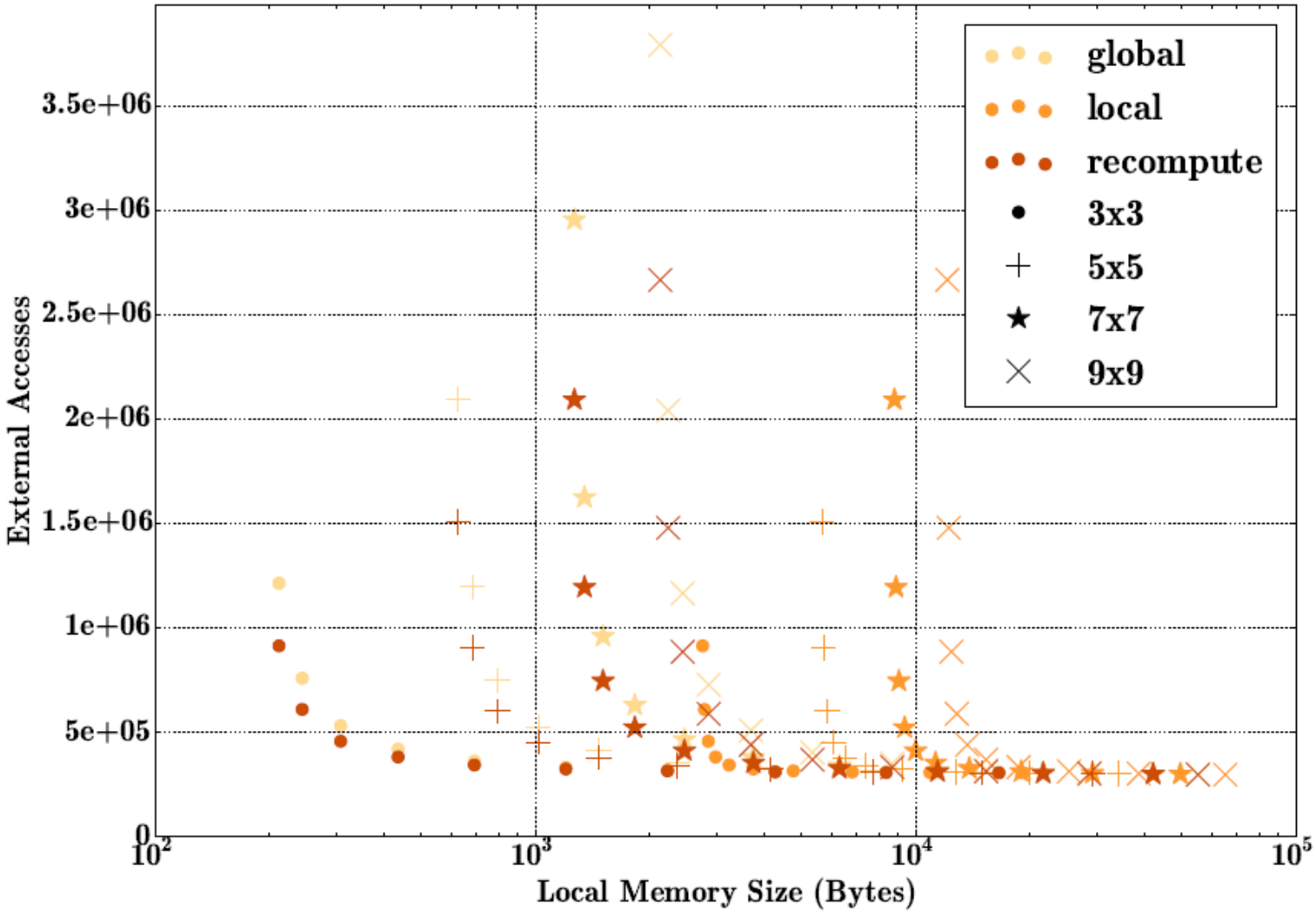
Results for different tile sizes



Results for different tile sizes



Results for different tile sizes and several kernel sizes



Technische Universiteit
Eindhoven
University of Technology

Conclusion and Future Work

Conclusion

- An example CNN application which includes recompute
- Extension of Polly
- Demonstration of the effectiveness of recomputation

Future Work

- Legality Checks
- Model of the effects
- More applications

And Finally…

- Questions?
- Remarks?
- Suggestions?