Lift Tutorial: Applications

Writing an Application

General Steps

- Determine input parameters
- Initialise input data
    - If testing, initialise comparison data
- Craft or translate the algorithm of interest
- Create an OpenCL kernel from your algorithm

Data Input to Lift Algorithms

- Lift can take in arrays or scalars as input parameters

```
1 val liftLambda = fun(
2   ArrayType(Float, SizeVar("N")),
3   ArrayType(Float, weights.length),
4   ...
5 )
```

- Single entry point for arrays into functions
  - Multiple arrays can be zipped together (but must be the same size!)

```
1 fun(neighbourhood) =>
2 {
3     ...
4     $ Zip(weights, neighbourhood)
5 }
```

Initialising Data in Scala

- Create arrays of data to pass into Lift algorithms in Scala

```
val stencilValues = Array.tabulate(nx,ny,nz) { (i,j,k) => (i + j + k + 1).toFloat }
```

- Our examples are all in unit tests, which include data to compare against - often from the same algorithm in Scala

```
assertEquals(dotProductScala(lift,right), output.sum, 0.0f)
```

Developing an Algorithm

The goal is not for Lift to be programmed in directly.

However, functionality for new types of algorithms must be added in and tested. In doing so, there are a few things to keep in mind:

- Lift allows multiple inputs, but there is only one data entry point to the main algorithm (can contain tuples)
- The algorithm itself must eventually map values back to global memory
- The result will be returned in a single array (however, this array can also contain tuples)

Simple Example: 1D Jacobi Stencil

```
1 val jacobi1Dstencil = fun(
2   ArrayType(Float, N),
3   (input) => {
4     Map(Reduce(add, 0.0f)) o
5       Slide(3, 1) o
6         Pad(1, 1, clamp) $ input
7   }
8 )
```

Creating an OpenCL kernel

- To compile your Lift kernel to OpenCL, run [opencl.executor]Compile(*<kernel>*)
  - This kernel can then be saved as a string or file

```
Compile(lambda)
```

- To execute the kernel straight away (compiling will happen behind the scenes), run [opencl.executor]Execute(*<options>*)[Array[*type*](*lambda*, *..inputs..*)

```
val (output, runtime) = Execute(inputData.length)[Array[Float]](stencilLambda, inputData, stencilWeights
```

Detailed Example: Matrix-Matrix Multiplication

Matrix-Matrix Multiplication Overview

- Widely used algorithm in mathematics, physics, engineering, etc
- Range of possible optimisations can be used, the performance of which varies across architectures
- Lift makes it easy to test these optimisations out without having to rewrite the original code

Matrix-Matrix Multiplication Example Code

```
1  val MMultiply = fun(
2    ArrayTypeWSWC(ArrayTypeWSWC(Float, K), M),
3    ArrayTypeWSWC(ArrayTypeWSWC(Float, K), N),
4    (matA, matB) => {
5      Map(fun( Arow =>
6        Map(fun( Bcol =>
7          Map(id) o Reduce(add, 0.0f) o Map(mult) $ Zip(Arow, Bco)
8        )) $ matB
9      )) $ matA
10   })
```

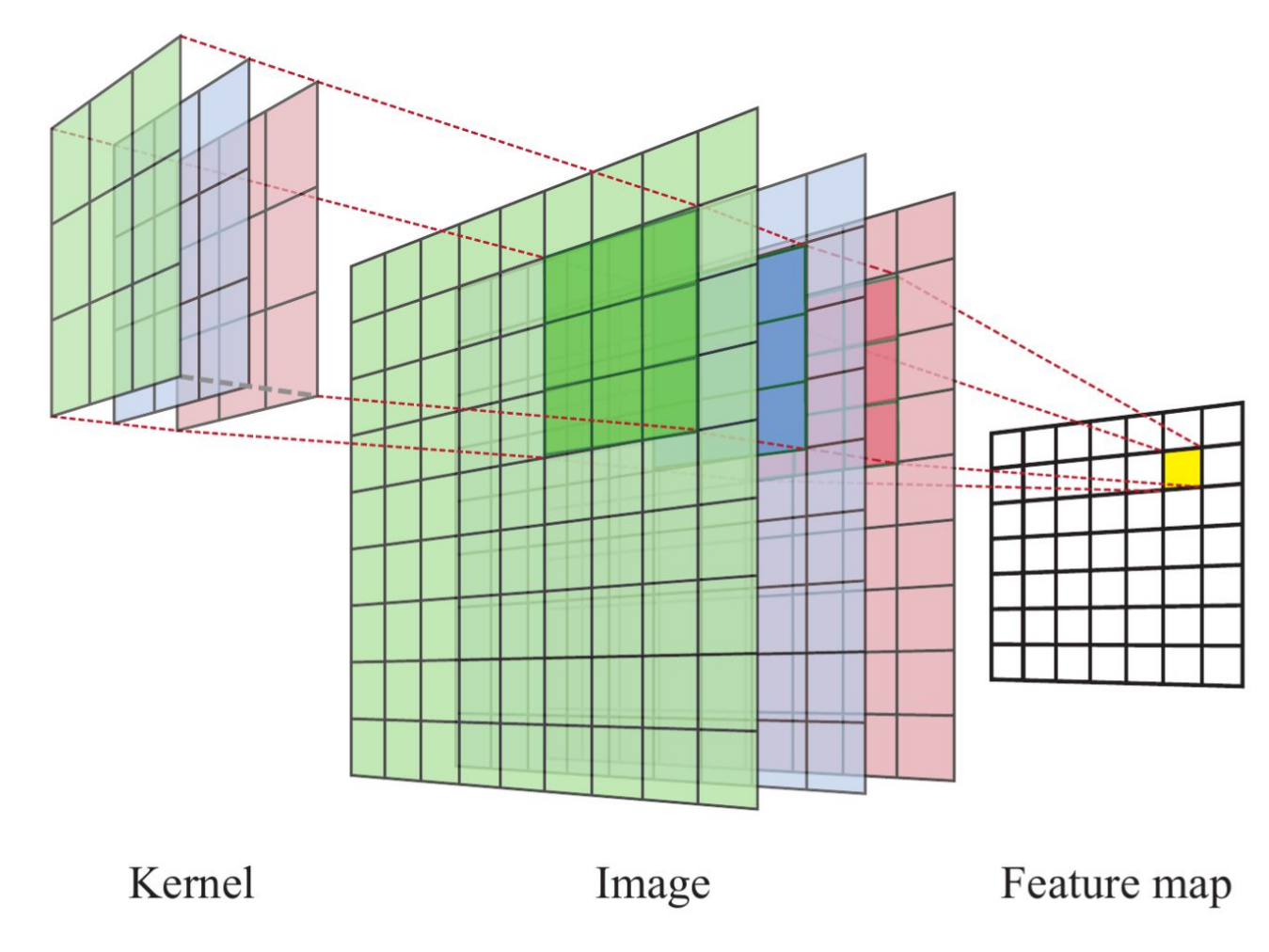
Matrix-Matrix Multiplication Using Rewrites

$$
\begin{aligned}
&\mathbf{A} * \mathbf{B} = \\
&\quad Map(\overrightarrow{rowA} \mapsto \\
&\qquad Map(\overrightarrow{colB} \mapsto \\
&\qquad\quad DotProduct(\overrightarrow{rowA}, \overrightarrow{colB}) \\
&\qquad) \circ Transpose() \,\$\, \mathbf{B} \\
&\quad) \,\$\, \mathbf{A}
\end{aligned}
$$

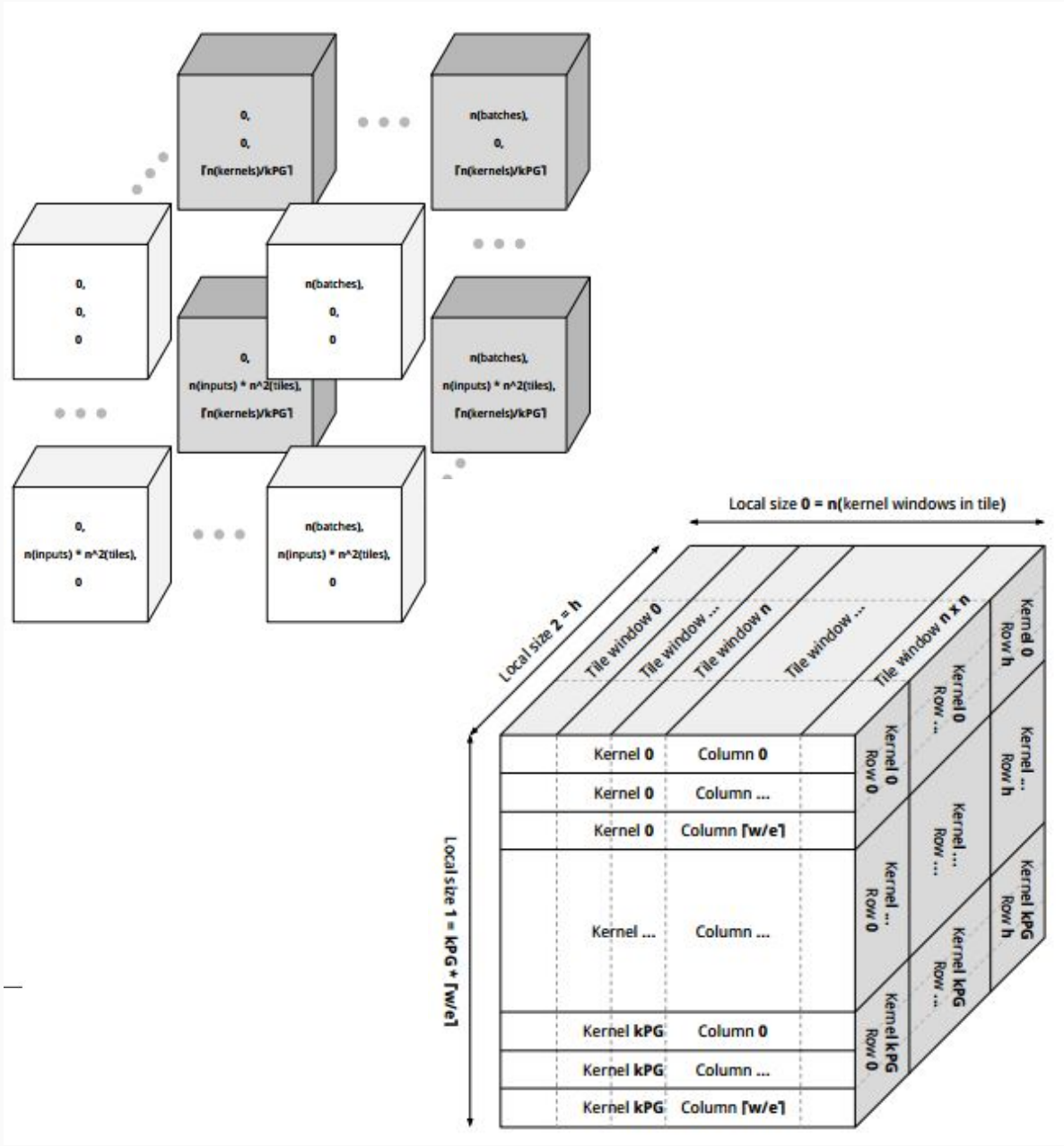
$$
\begin{aligned}
&(p239, p36 \mapsto \\
&\quad Join() \circ Map((p179 \mapsto \\
&\qquad Transpose() \circ Join() \circ Map((p70 \mapsto \\
&\qquad\quad Transpose() \circ Join() \circ Map((p20 \mapsto \\
&\qquad\qquad Transpose() \circ Map((p65 \mapsto \\
&\qquad\qquad\quad Transpose()(p65) \\
&\qquad\qquad)) \circ Transpose()(p20) \\
&\qquad\quad)) \circ Transpose() \circ Reduce((p75, p0 \mapsto \\
&\qquad\qquad Map((p164 \mapsto \\
&\qquad\qquad\quad Join() \circ Map((p81 \mapsto \\
&\qquad\qquad\qquad Reduce((p136, p90 \mapsto \\
&\qquad\qquad\qquad\quad Map((p163 \mapsto \\
&\qquad\qquad\qquad\qquad Get(0)(p163) + Get(1)(p163) * Get(1)(p90) \\
&\qquad\qquad\qquad\quad)) \circ Zip(2)(p136, Get(0)(p90)) \\
&\qquad\qquad\qquad))(Get(0)(p81), Zip(2)(Transpose() \circ Get(1)(p164), Get(1)(p81))) \\
&\qquad\qquad\quad)) \circ Zip(2)(Get(0)(p164), Get(1)(p0)) \\
&\qquad\qquad)) \circ Zip(2)(p75, Split(blockFactor) \circ Transpose() \circ Get(0)(p0)) \\
&\qquad\quad))(Zip(2)(Split(sizeK) \circ Transpose()(p179), p70)) \\
&\qquad)) \circ Transpose() \circ Map((p4 \mapsto \\
&\qquad\quad Split(sizeN) \circ Transpose()(p4) \\
&\qquad)) \circ Split(sizeK)(p36) \\
&\quad)) \circ Split(sizeM)(p239)
\end{aligned}
$$

Detailed Example: Convolutional Neural Network

Convolutional Neural Network – Overview



Convolutional Neural Network – OpenCL ND space



Detailed Example: Acoustics Simulation

Acoustics Simulation Overview

- Partial Differential Equations can be discritised into stencils to model physical simulations like 3D wave models
- Room acoustics simulations have been explored extensively in Lift

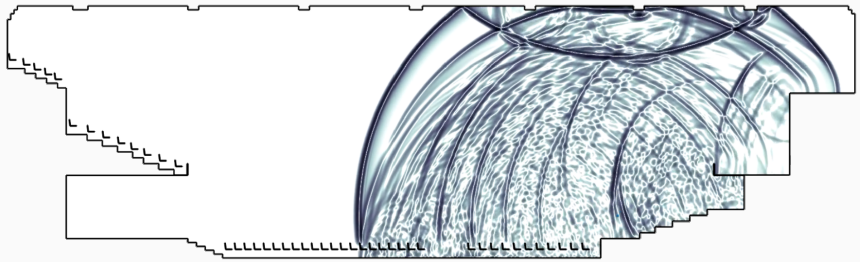


Figure 1: Cross Section of a 3D Concert Hall Model