Hewlett Packard Enterprise

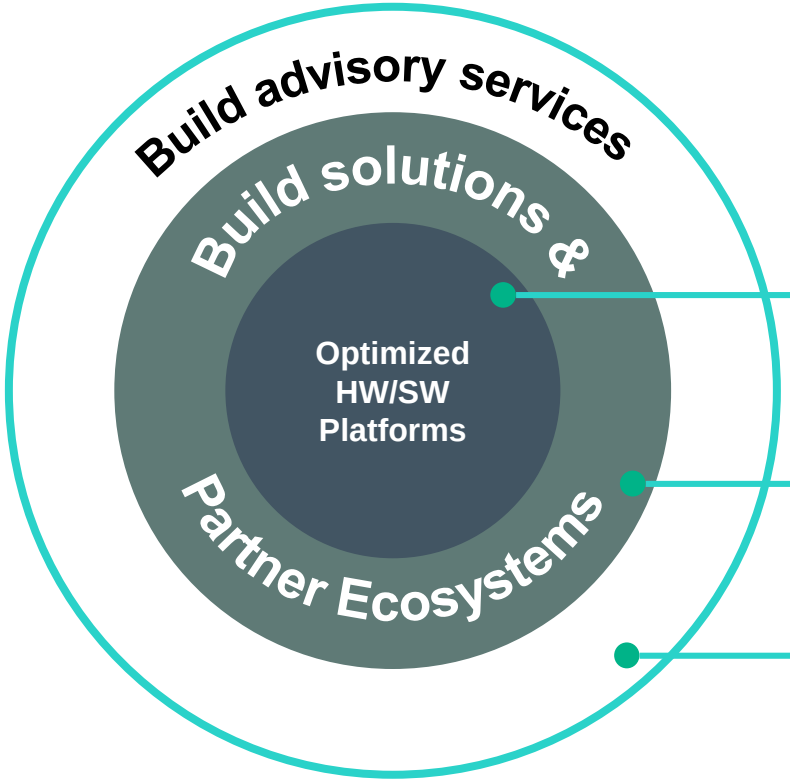
A Platform for Accelerating Machine Learning Applications

Ben Chandler
Hewlett Packard Labs

April 6th, 2016

HPE Big Data and HPC portfolio strategy

Design and deliver comprehensive solutions with purpose-built platforms



1. **Innovate, design & deliver the best-in-class hardware and software** to support foundational infrastructure needs of the Big Data customers
2. **Provide vertical solutions** by building software stack and partner ecosystem
3. Enable Advisory Services to help **manage customer's technology journey**

Drive HPC and Big Data across all Enterprises

Modernize your datacenter for massive parallel processing innovation

Deliver automated intelligence, real-time insights and optimized performance

Navigate the data-driven transformation journey across all enterprises with new HPC and Big Data capabilities that accelerate time-to-value for increased competitive differentiation

Automated intelligence	Real-time insights			Optimized performance		
Deep Learning Innovation	HPC Compute & Storage Solution	HPE Vertica for SQL on Hadoop	Integrity MC990 X for Database Processing	Risk Compliant Archive Solution	Trade & Match Server Solution	HPC for Trader Workstation
Apollo 6500, Apollo 4520		Apollo 4000 Series		Apollo 4510	Apollo 2000	HPE Moonshot

Extreme performance capabilities to process, manage and analyze data, I/O and storage intensive application workloads with high speed, scale, efficiency and enable high flexibility for open infrastructure innovation

Deliver automated intelligence in real-time for Deep Learning

Unprecedented performance and scale with HPE Apollo 6500 high density GPU solution

Use Cases

Automated Intelligence
delivered by HPE Apollo 6500 and Deep Learning software solutions

Video, Image, Text, Audio, time series pattern recognition	Large, highly complex, unstructured simulation and modeling	Real-time, near real-time analytics

Faster Model training time, better fusion of data*

Customer benefits

HPE Apollo 6500 is an ideal HPC and Deep Learning platform providing **unprecedented performance with 8 GPUs, high bandwidth fabric and a configurable GPU topology to match deep learning workloads**

- Up to 8 high powered GPUs per tray (node), 2P Intel E5-2600 v4 support
- Choice of high-speed, low latency fabrics with 2x IO expansion
- Workload optimized using flexible configuration capabilities

* Benchmarking results provided at or shortly after announcement

HPE Apollo 6500 solution innovation

System Design Innovation to maximize GPU capacity and performance with lower TCO

New technologies, products

Deep Learning, HPC Software platform Enablement
(HPE CCTK, Caffe, CUDA, Google TensorFlow, HPE IDOL)

Cluster Management Enhancements
(Massive Scaling, Open APIs, tight Integration, multiple user interfaces)

HPE Apollo 6500

NEW

- Dense GPU server optimized for Deep Learning and HPC workloads
- Density optimization
- High performance fabrics

Unique Solution differentiators

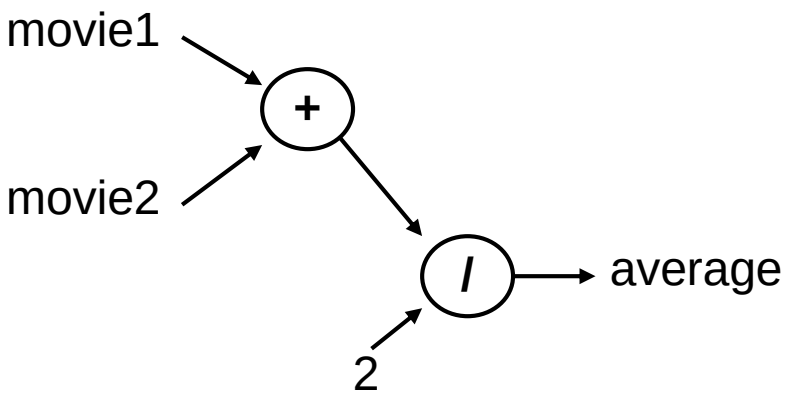
- GPU density
- Configurable GPU topologies
- More network bandwidth
- Power and cooling optimization
- Manageability
- Better productivity

Roadmap

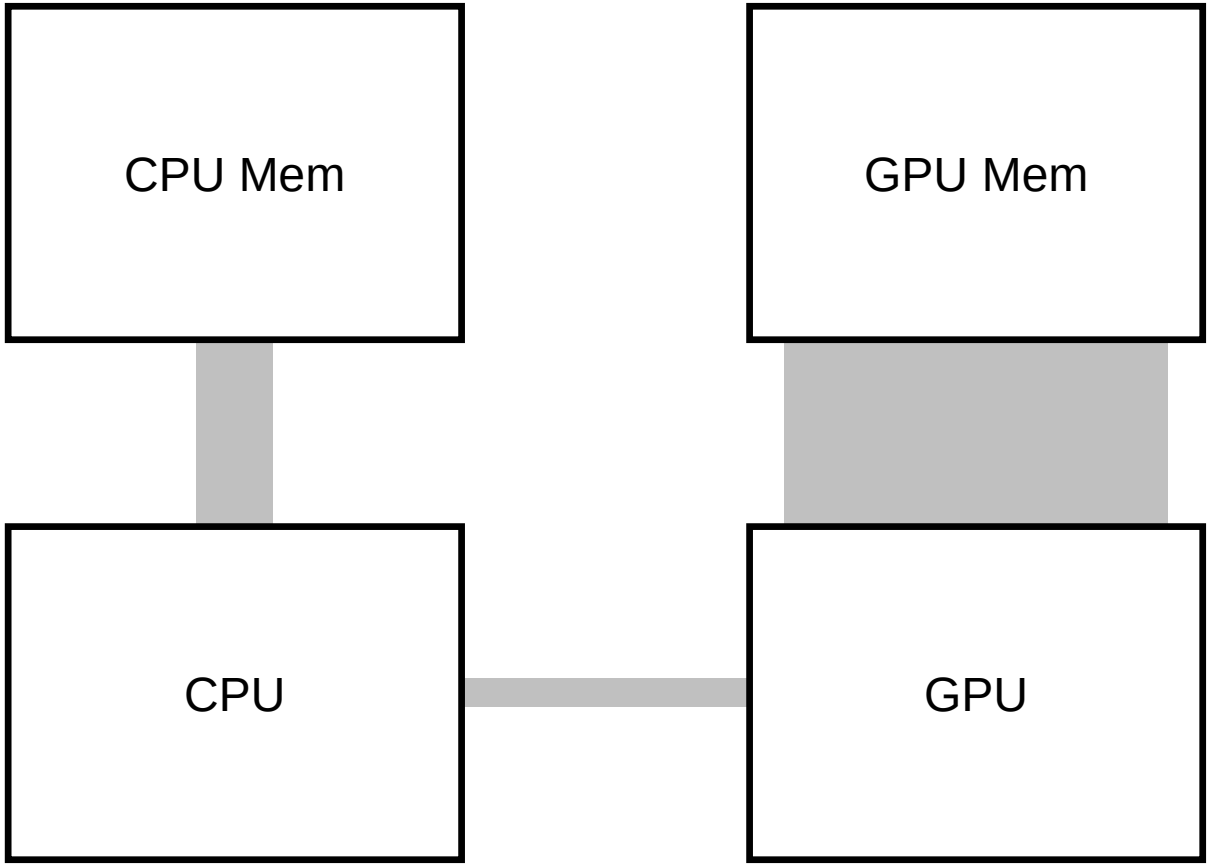
- **Motivating evidence**
- The CogX project and vision
- Open-source availability

A simple data-intensive program

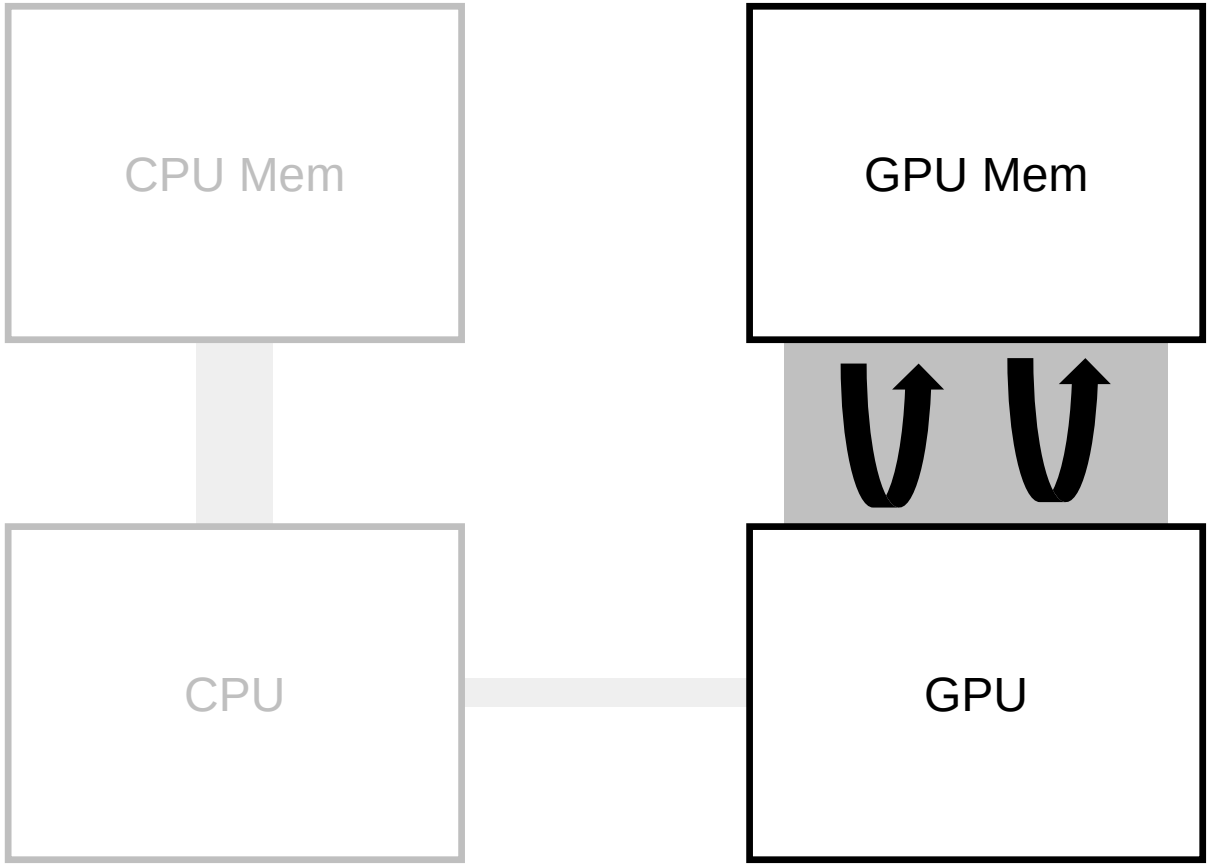
```
val movie1 = ...
val movie2 = ...

val average = (movie1 + movie2) / 2
```



Simplified architecture diagram

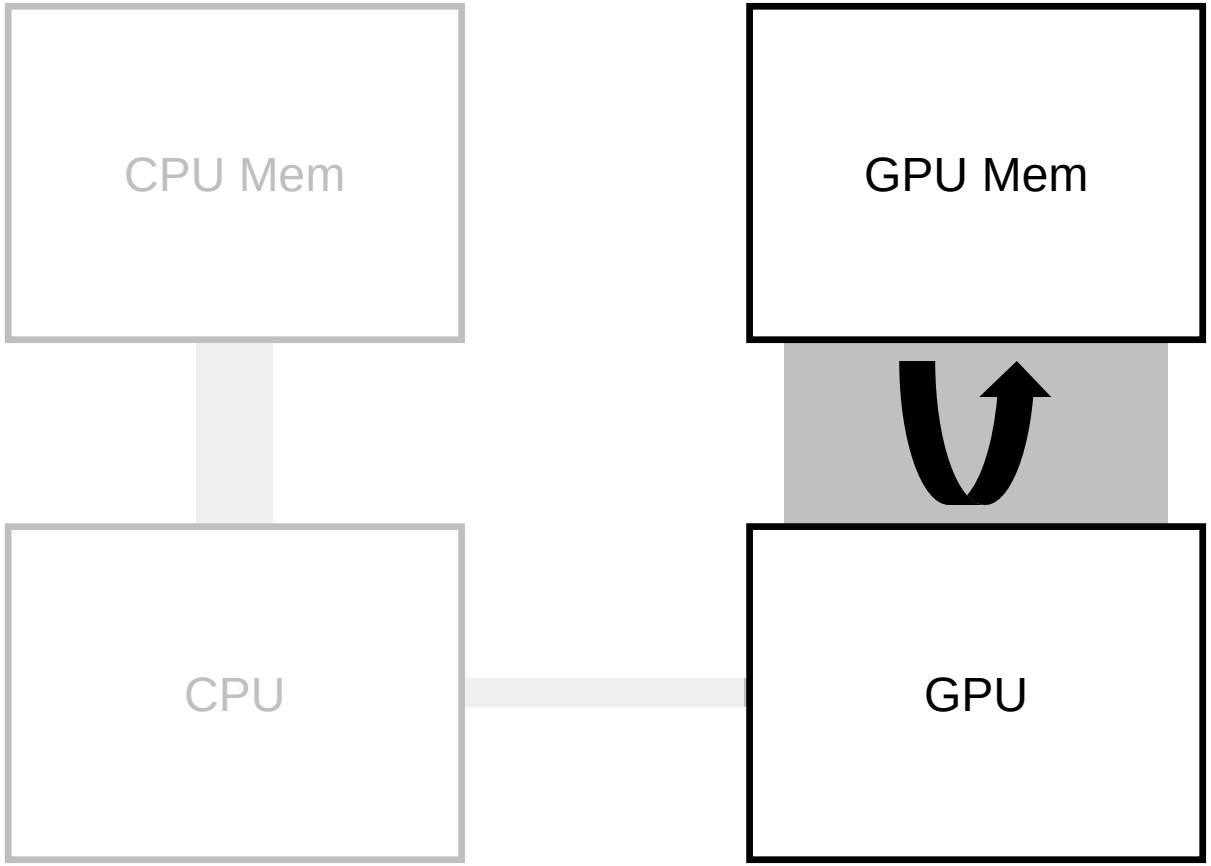


Naïve data flow in practice



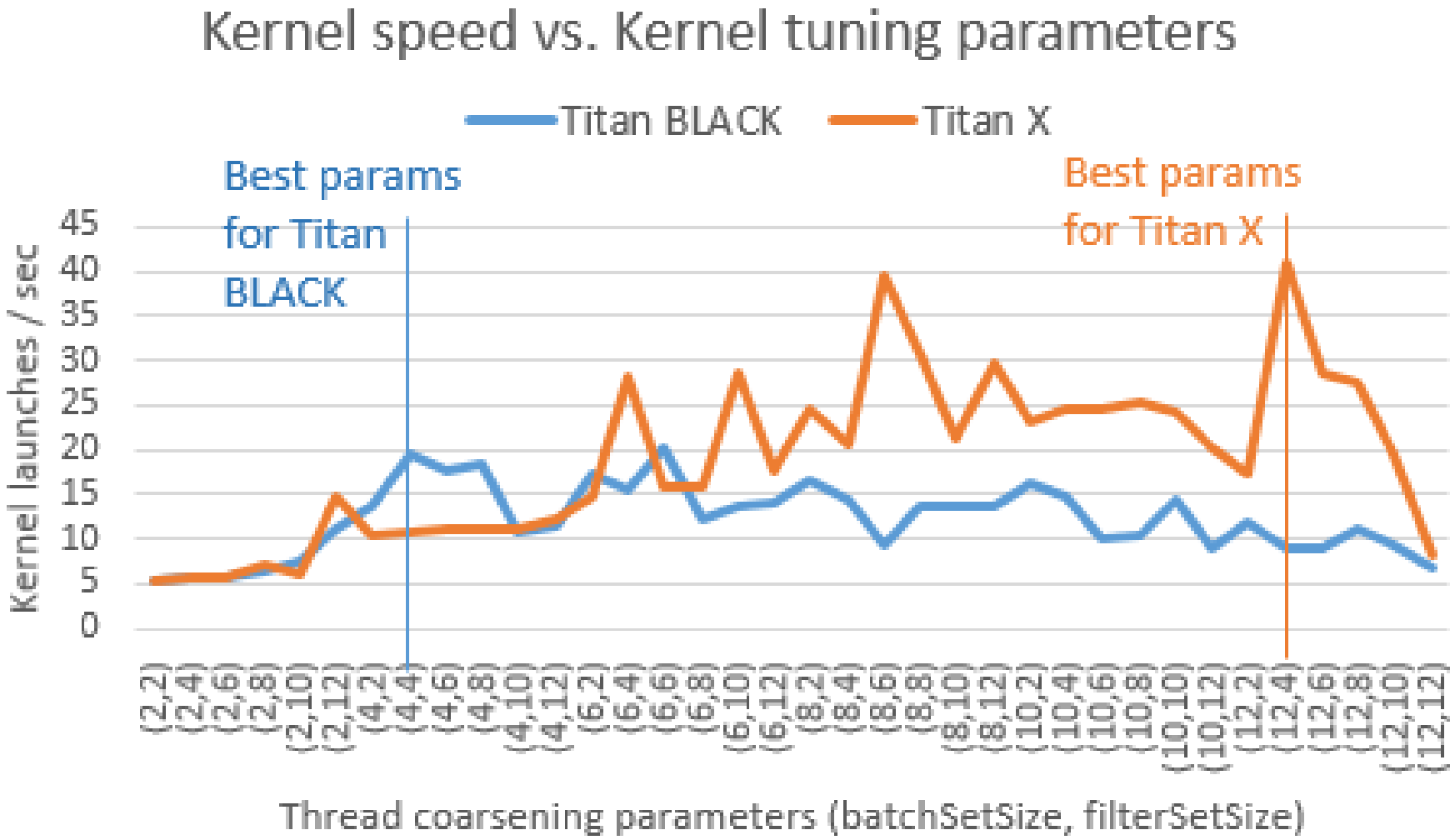

```
val average = (movie1 + movie2) / 2
```

Optimized data flow in practice




```
val average = fusedOp(movie1, movie2, 2)
```

Performance portability on GPUs



Roadmap

–Motivating evidence

–The CogX project and vision

–Open-source availability

Vision

performance-portable, high-productivity programming for accelerators

CogX

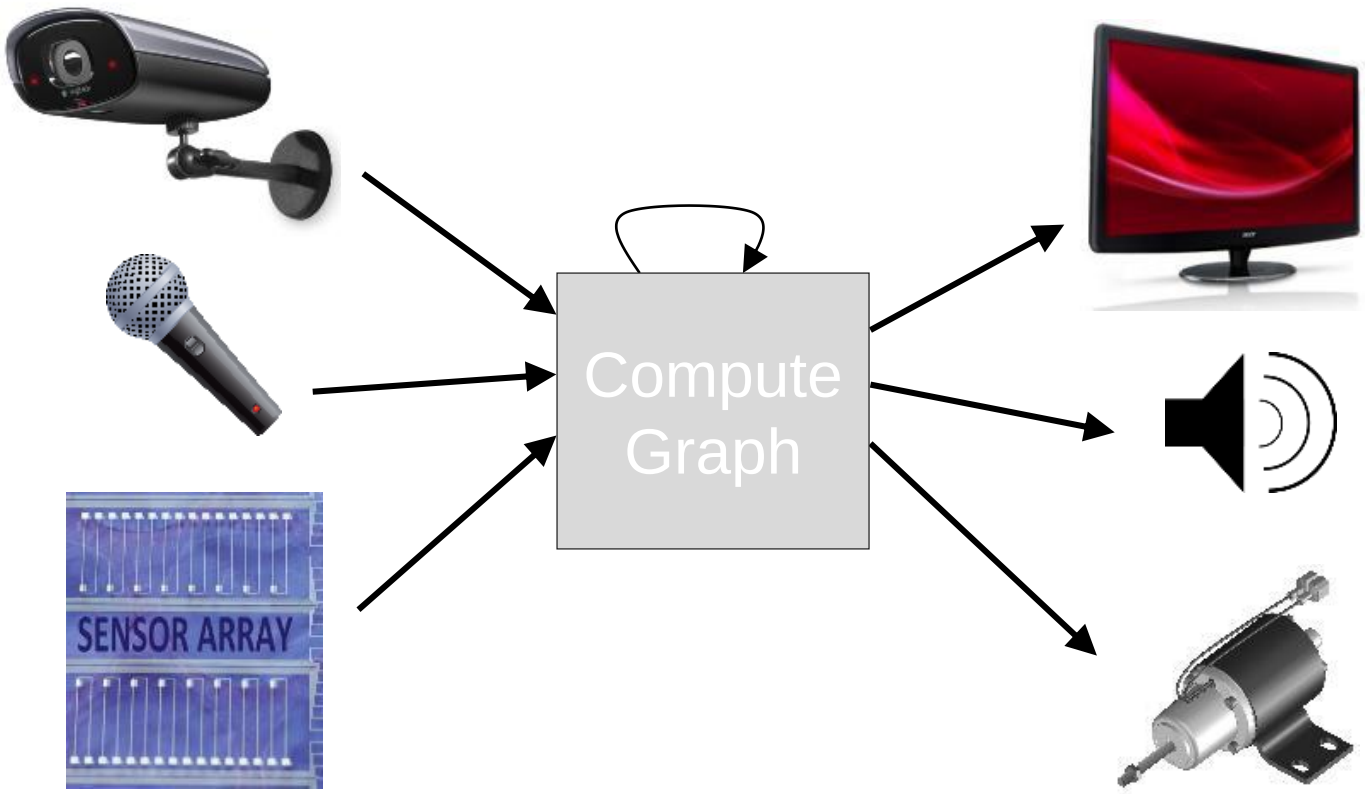
What is CogX?

- Domain-specific embedded language with associated optimizing compiler and runtime
- Array programming language embedded in a state machine execution model
- Targets advanced analytics workloads on massively parallel distributed systems

- Design Goals
  - Optimal deployment on parallel hardware
  - Fast design iterations
  - Enforce scalability
  - Broad COTS hardware support
  - Compatible with shared infrastructure
  - **High productivity for analysts and algorithm engineers**

CogX compute model

- Compute Graphs
  - Fields
  - Operators
  - Sensors/Actuators
  - Feedback/Time



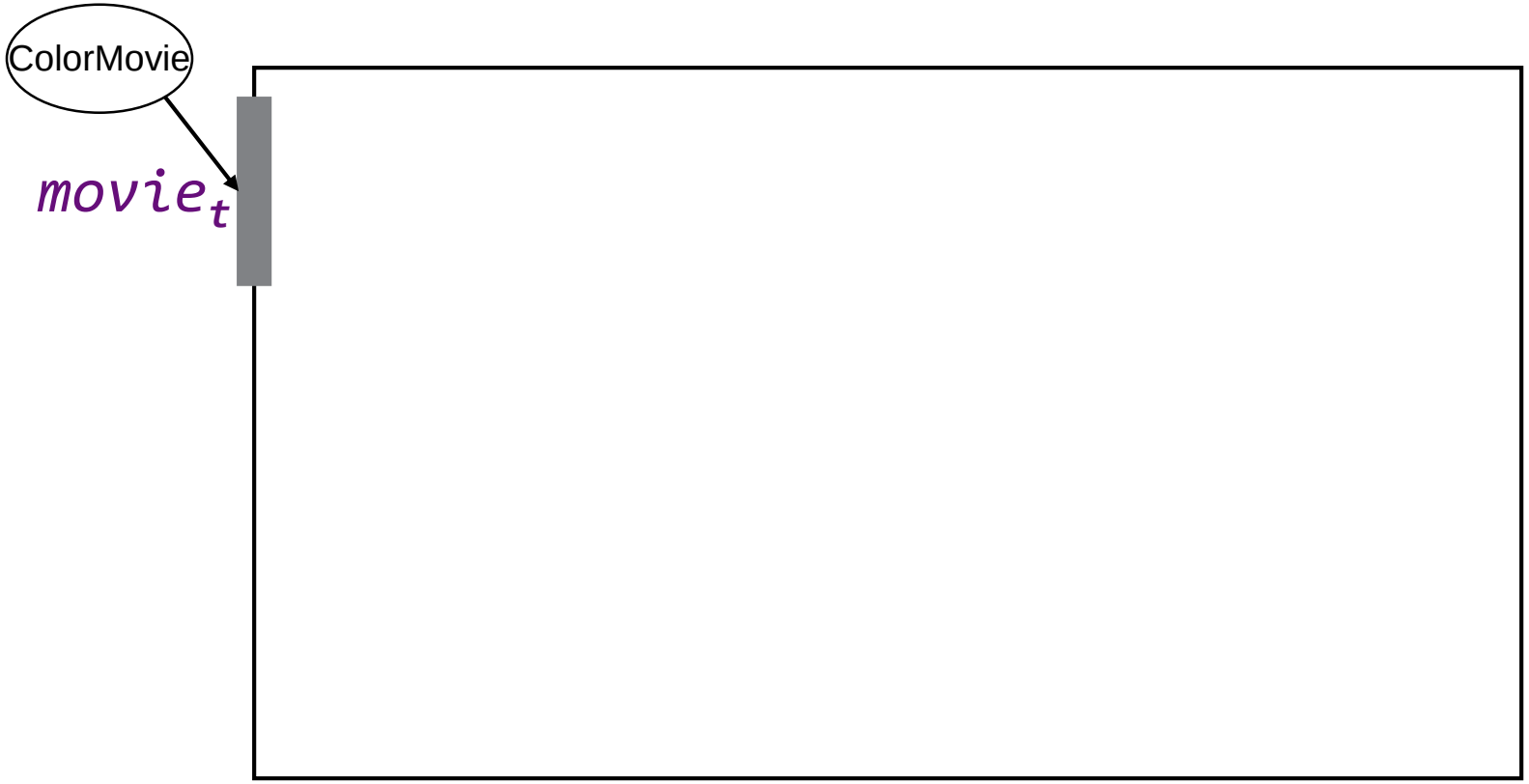
CogX compute model

```
val movie = ColorMovie("courtyard.mp4")
val background = VectorField(movie.fieldShape, Shape(3))
val nextBackground = 0.999f * background + 0.001f * movie
background <== nextBackground
val suspicious = reduceSum(abs(movie - background))
```

Demo: Hello World application

CogX compute model

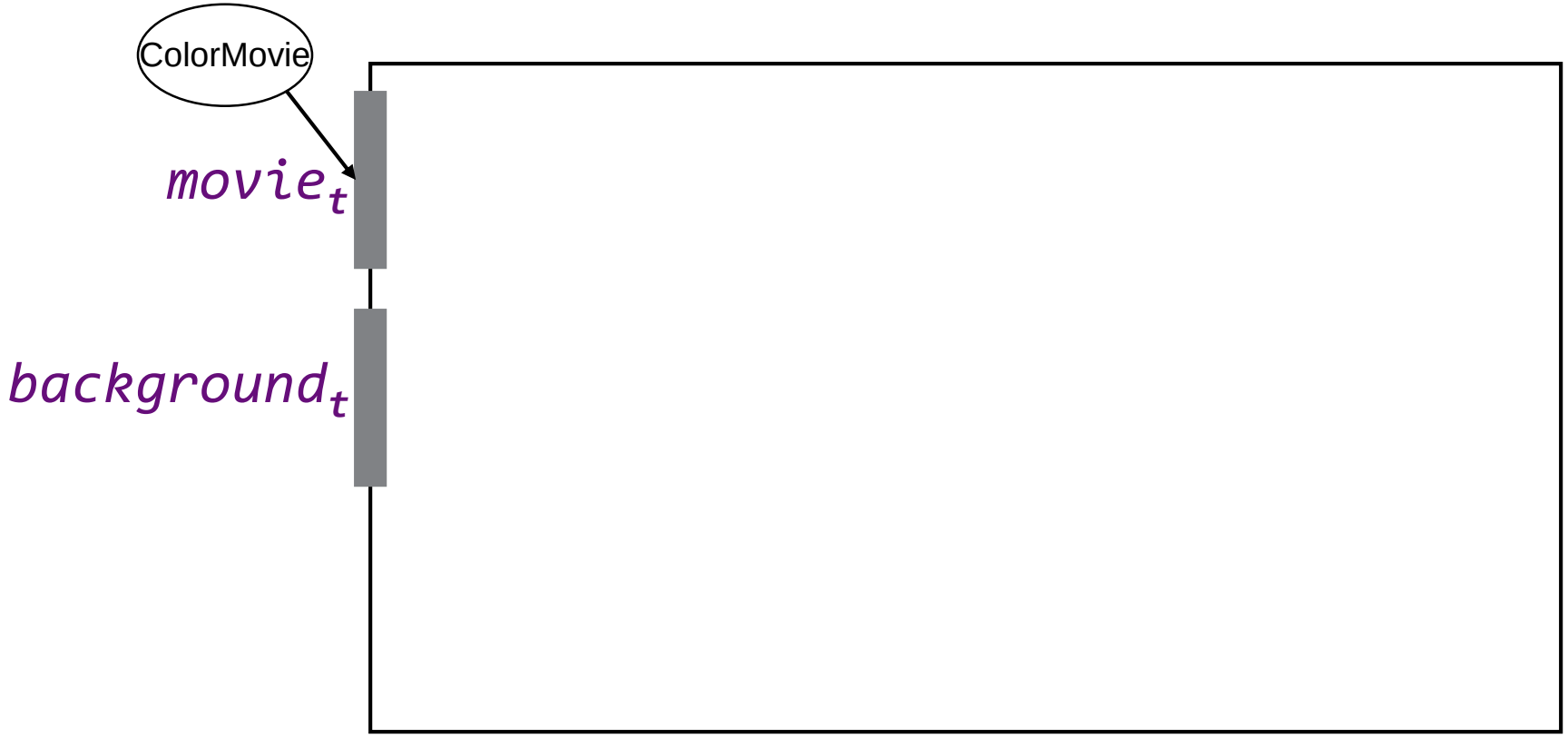
```
val movie = ColorMovie("courtyard.mp4")
```



Compute graph

CogX compute model

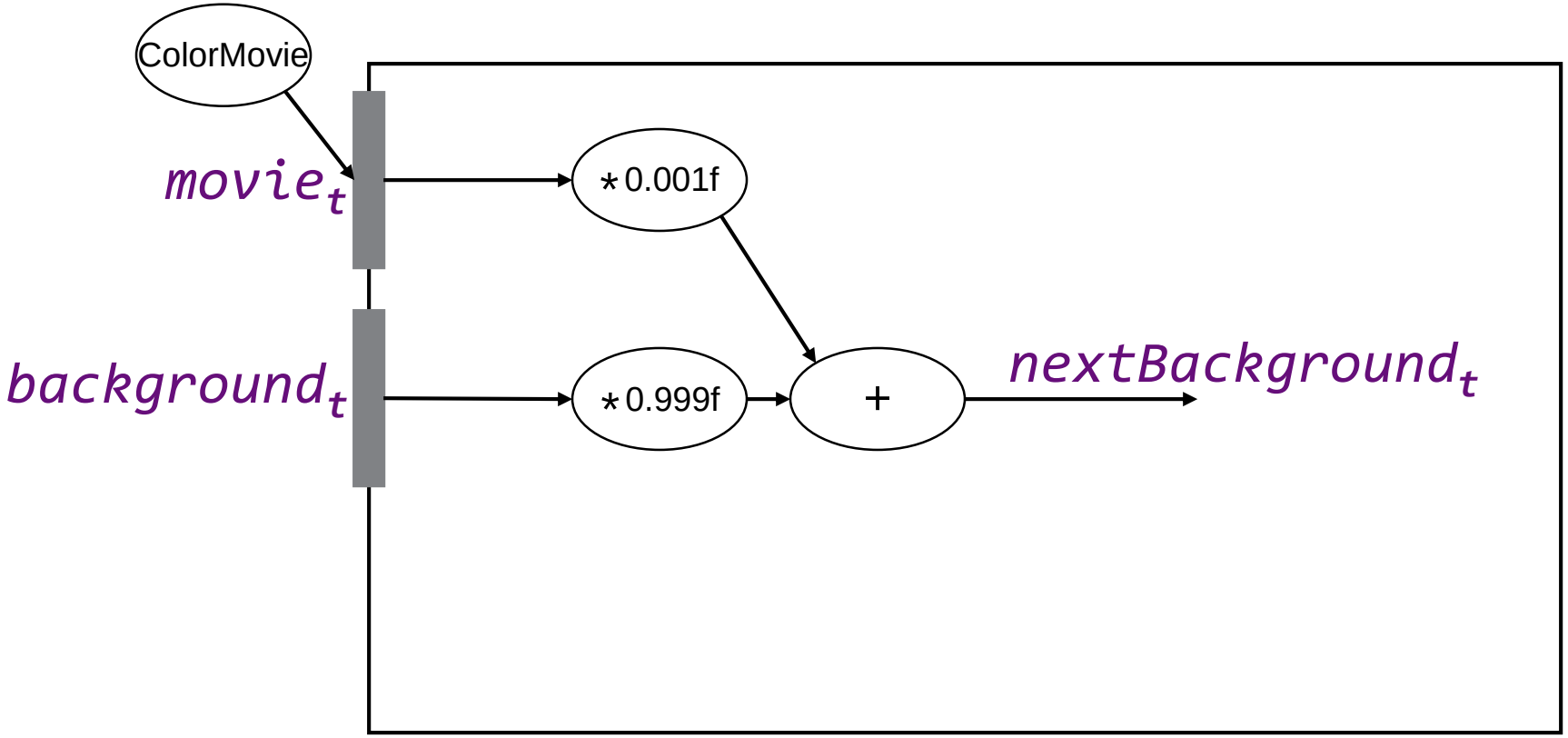
```
val background = VectorField(movie.fieldShape, Shape(3))
```



Compute graph

CogX compute model

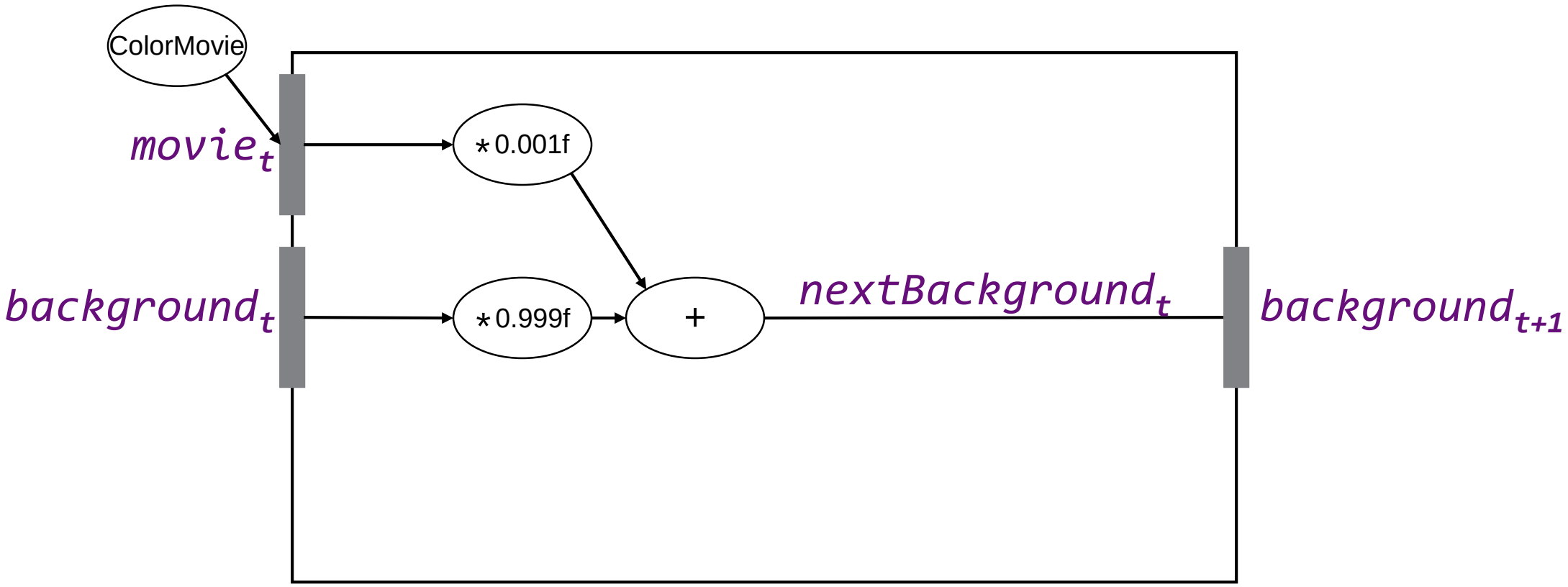
```
val nextBackground = 0.999f * background + 0.001f * movie
```



Compute graph

CogX compute model

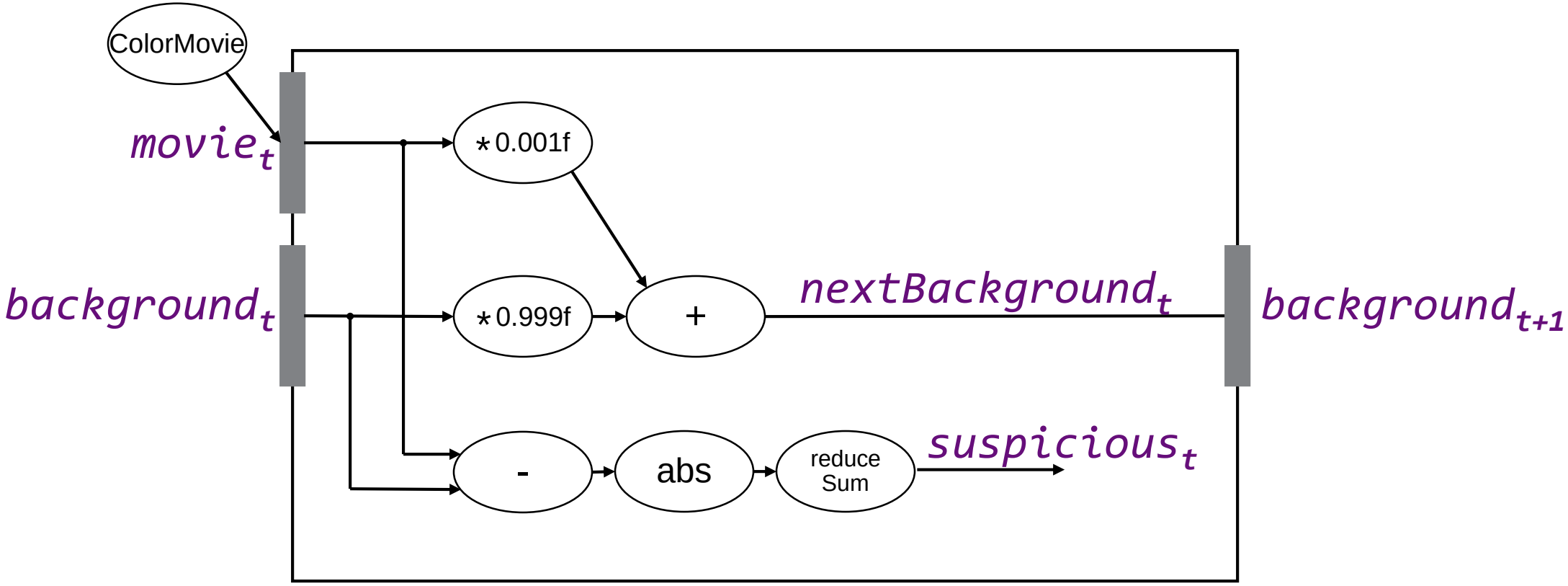
background <== *nextBackground*



Compute graph

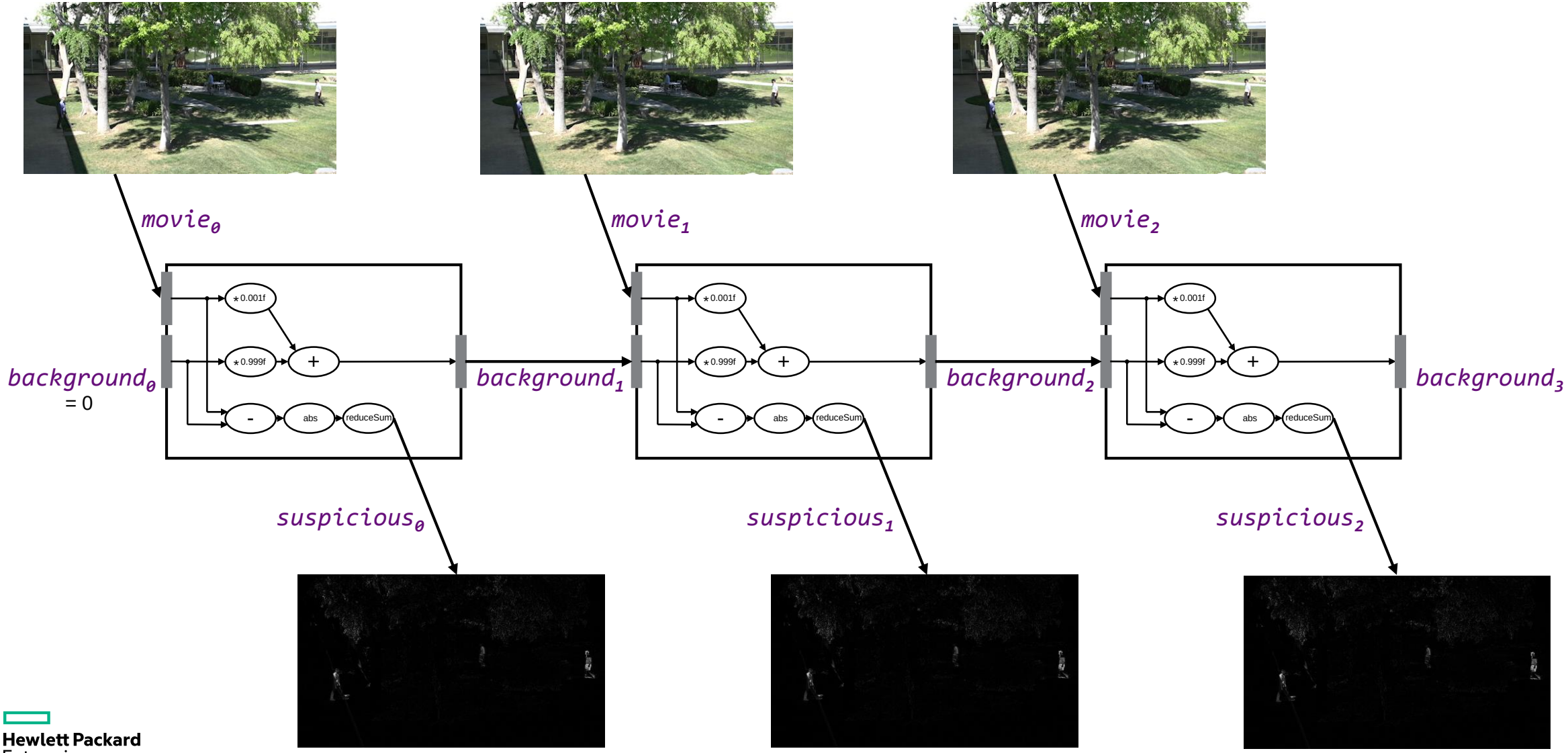
CogX compute model

```
val suspicious = reduceSum(abs(movie - background))
```



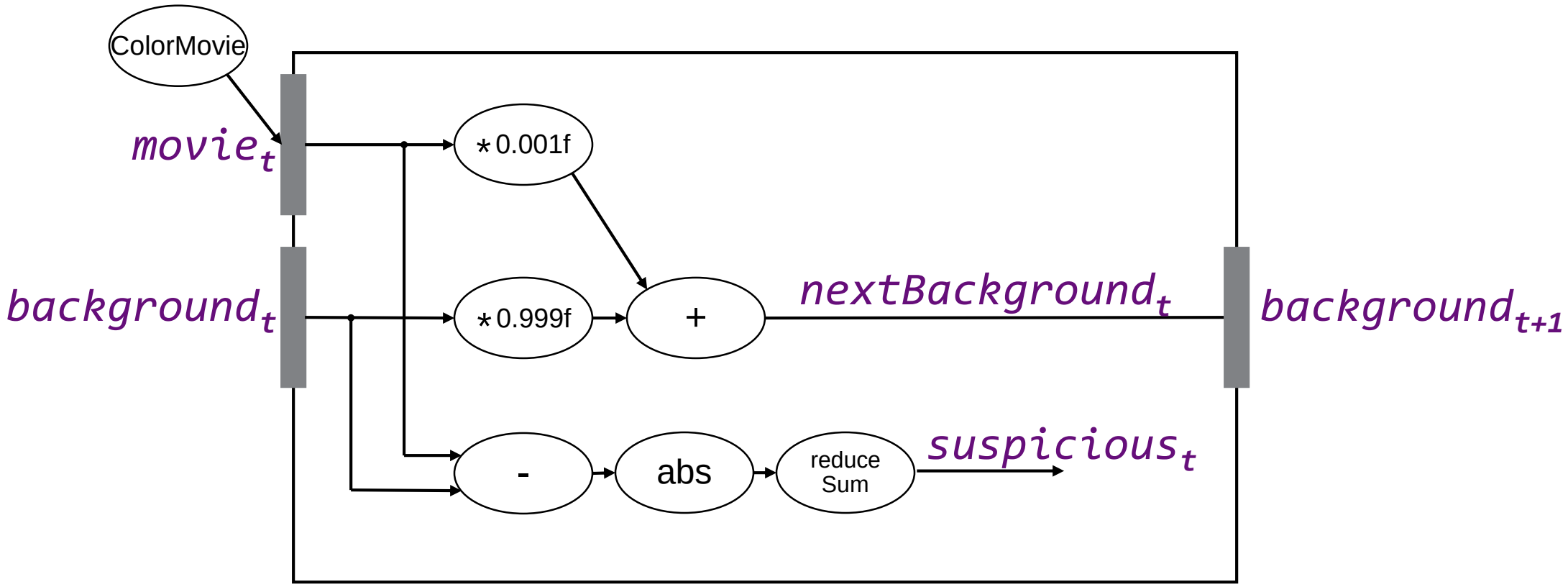
Compute graph

CogX compute model



Hewlett Packard
Enterprise

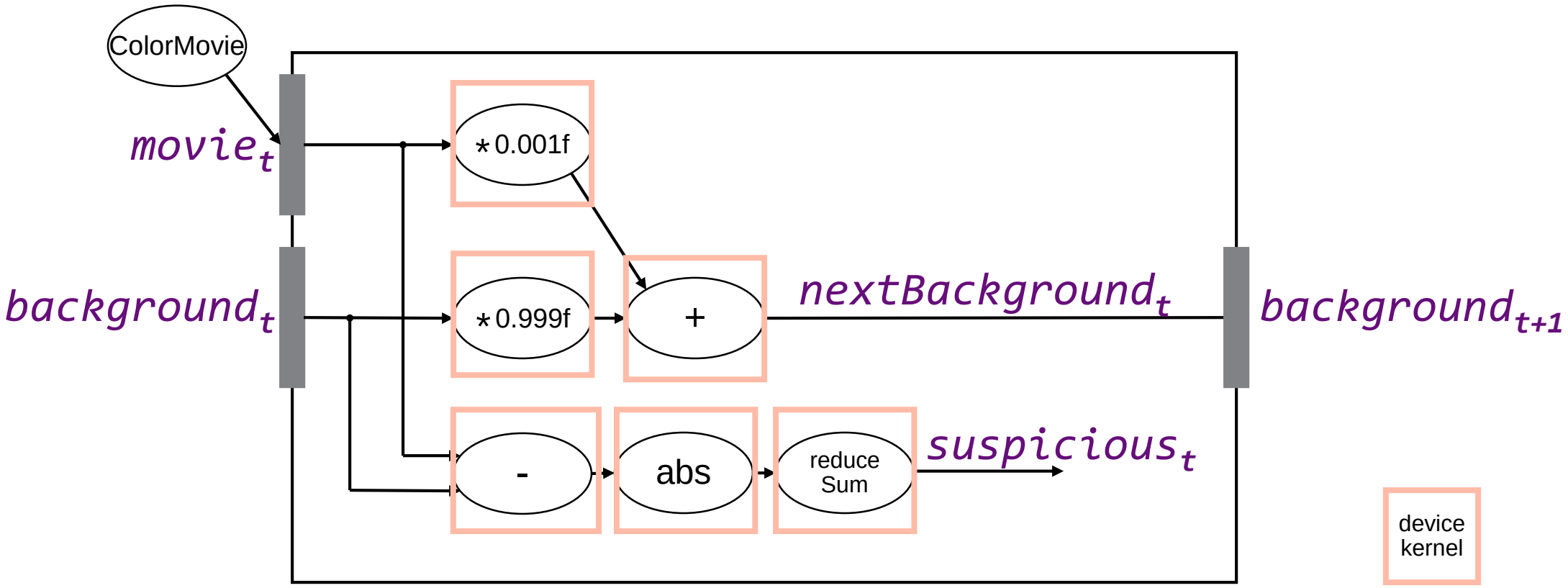
Opportunities for optimization



Compute graph

Opportunities for optimization

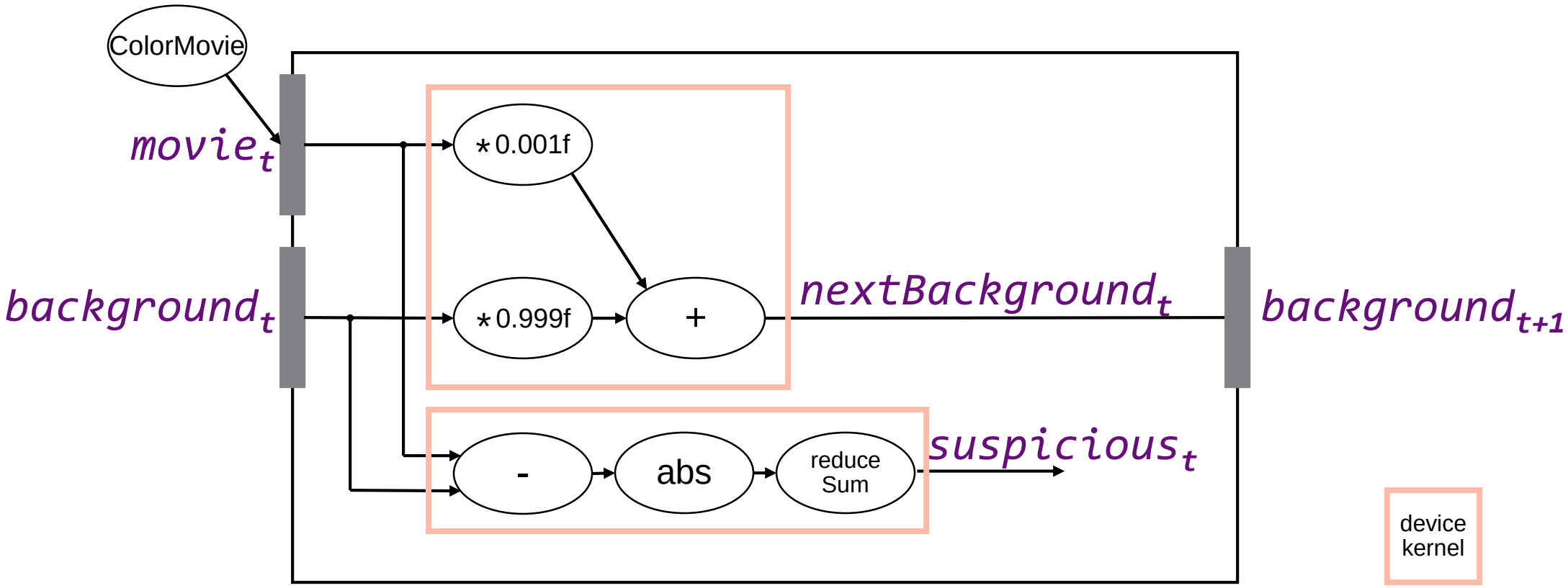
Initially: 6 separate device kernels.



Compute graph

Opportunities for optimization

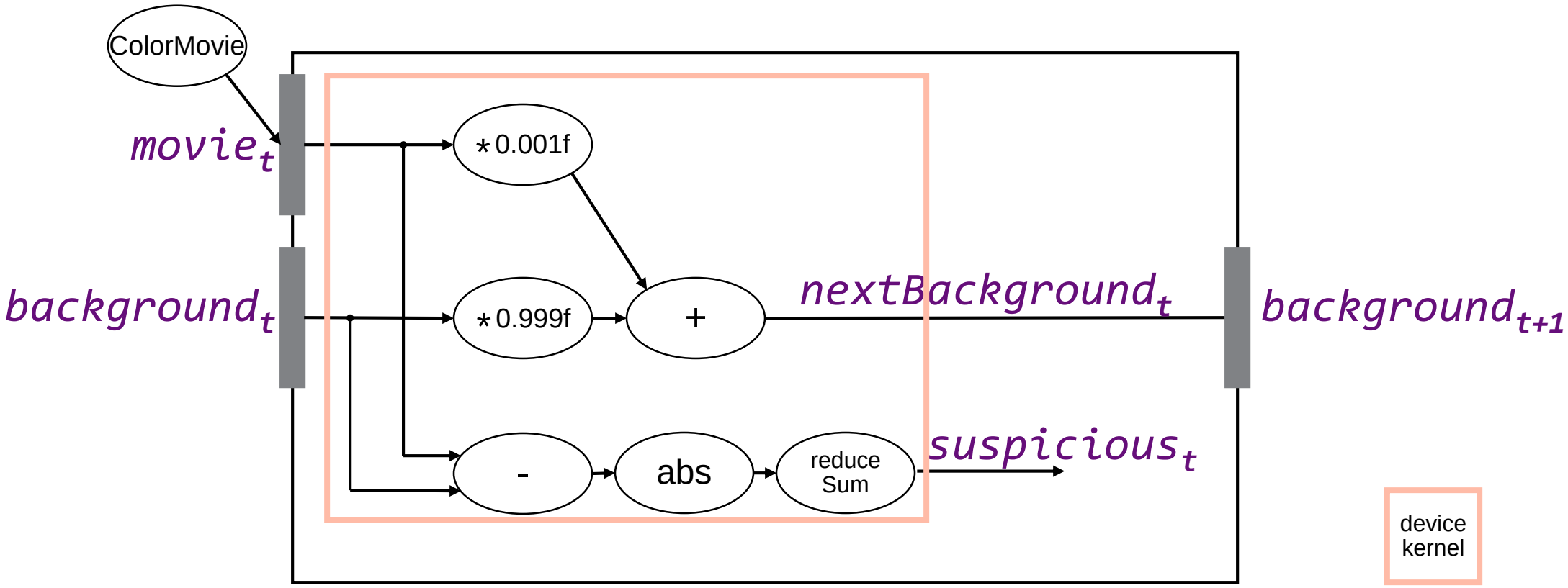
After a “single-output” kernel fuser pass: 2 device kernels remain.



Compute graph

Opportunities for optimization

After a “multi-output” kernel fuser pass: only a single device kernel remains.



Compute graph

CogX compiler: translating CogX to OpenCL with kernel fusion

User CogX model

(scala)

parsing and OpenCL code generation

Kernel circuit

(kernels, field bufs)

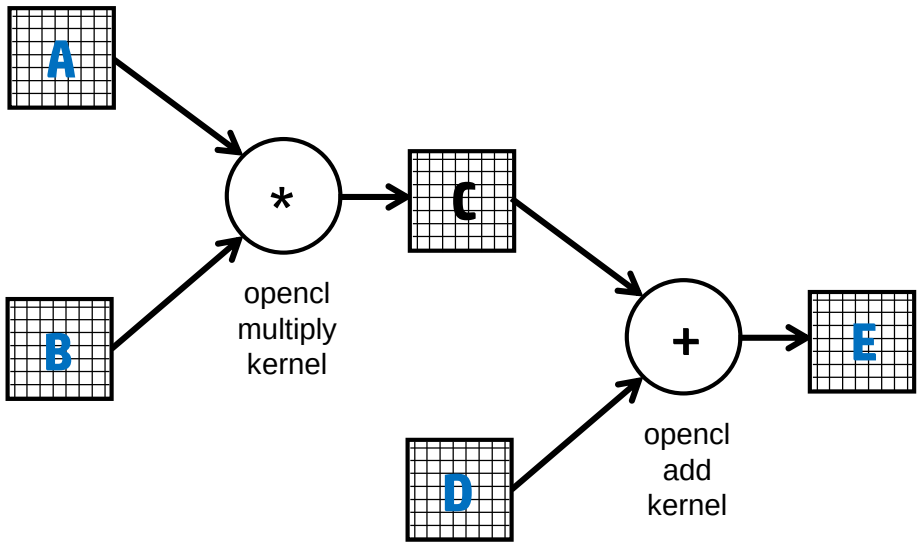
optimizations, including kernel fusion

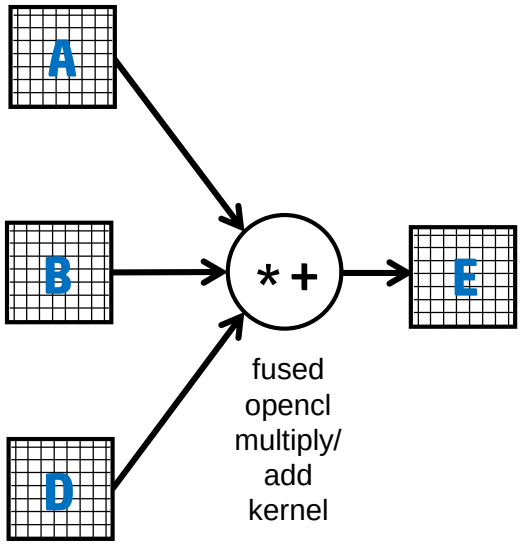
Optimized kernel circuit

(merged kernels)

CogX code snippet

```
val A = ScalarField(10,10)
val B = ScalarField(10,10)
val C = A * B
val D = ScalarField(10,10)
val E = C + D
```

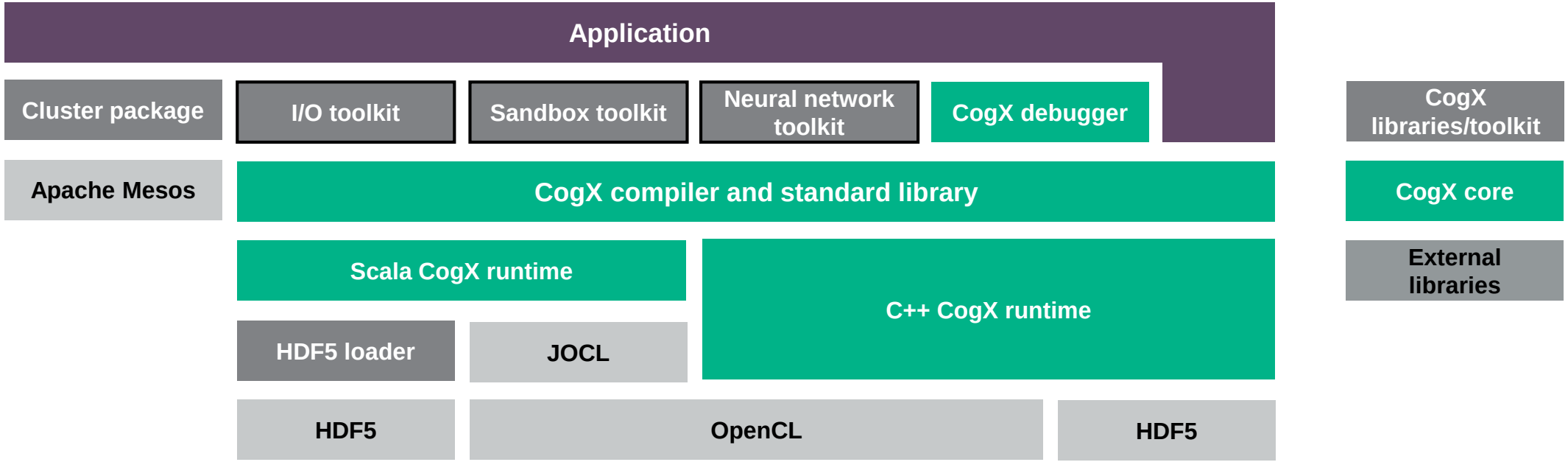




CogX core functions and operators

- Basic operators
  - +, -, *, /, %
- Logical operators
  - >, >=, <, <=, ===, !===
- Pointwise functions
  - cos, cosh, acos
  - sin, sinh, asin
  - tan, tanh, atan2
  - sq, sqrt, log, signum
  - pow, reciprocal
  - exp, abs, floor
- Comparison functions
  - max, min
- Shape manipulation
  - flip, shift, shiftCyclic
  - transpose, subfield
  - expand, select, stack
  - matrixRow, reshape
  - subfields, trim
  - vectorElement, vectorElements
  - transposeMatrices
  - transposeVectors
  - replicate, slice
- FFT/DCT
  - fft, fftInverse
  - fftRI, fftInverseRI
  - fftRows, fftInverseRows
  - fftColumns, fftInverseColumns
  - dct, dctInverse, dctTransposed
  - dctInverseTransposed
- Complex numbers
  - phase, magnitude, conjugate
  - realPart, imaginaryPart
- Convolution-like
  - crossCorrelate, crossCorrelateSeparable
  - convolve, convolveSeparable
  - projectFrame, backProjectFrame
  - crossCorrelateFilterAdjoint
  - convolveFilterAdjoint
- Gradient/divergence
  - backwardDivergence
  - backwardGradient
  - centralGradient
  - forwardGradient
- Linear algebra
  - dot, crossDot
  - reverseCrossDot
- Debugging
  - probe
- Type coercion
  - toScalarField, toVectorField
  - toMatrixField, toComplexField
  - toComplexVectorField, toColorField
  - toGenericComplexField
- Type construction
  - complex, polarComplex
  - vectorField, complexVectorField
  - matrixField, colorField
- Reductions
  - reduceSum, blockReduceSum
  - reduceMin, blockReduceMin
  - reduceMax, blockReduceMax
  - fieldReduceMax, fieldReduceMin
  - fieldReduceSum, fieldReduceMedian
- Normalizations
  - normalizeL1, normalizeL2
- Resampling
  - supersample, downsample, upsample
- Special operators
  - winnerTakeAll
  - random
  - solve
  - transform
  - warp
  - <==

CogX software stack



Applications are written by users

- Introductory and training examples for single-GPU and distributed computation
- Performance benchmarks covering the core and neural network package
- Several larger-scale demo applications integrating multiple CogX functions

CogX toolkit functions

- Computer Vision
  - Annotation tools
  - Color space transformations
  - Polynomial dense optic flow
  - Segmentation
- Solvers
  - Boundary-gated nonlinear diffusion
  - FISTA solver (with sub-variants)
  - Golden section solver
  - Incremental *k*-means implementation
  - LSQR solver (with sub-variants)
  - Poisson solver (with sub-variants)
- Filtering
  - Contourlets
  - 4 frequency-domain filters
  - Mathematical morphology operators
  - 27 space-domain filters (from a simple box filter up to local polynomial expansion and steerable Gabor filters)
  - Steerable pyramid filter
  - Wavelets
  - Variants of whitening transforms
  - Contrast normalization
  - Domain transfer filter
  - Gaussian pyramid
  - Monogenic phase congruency
- Dynamical Systems
  - Kalman filter
  - Linear system modeling support
  - CPU matrix pseudo-inverse
- Statistics
  - Normal and uniform distributions
  - Histograms
  - Moment calculations
  - Pseudo-random number generator sensors

Labeling Dynamic Ordinal Depth

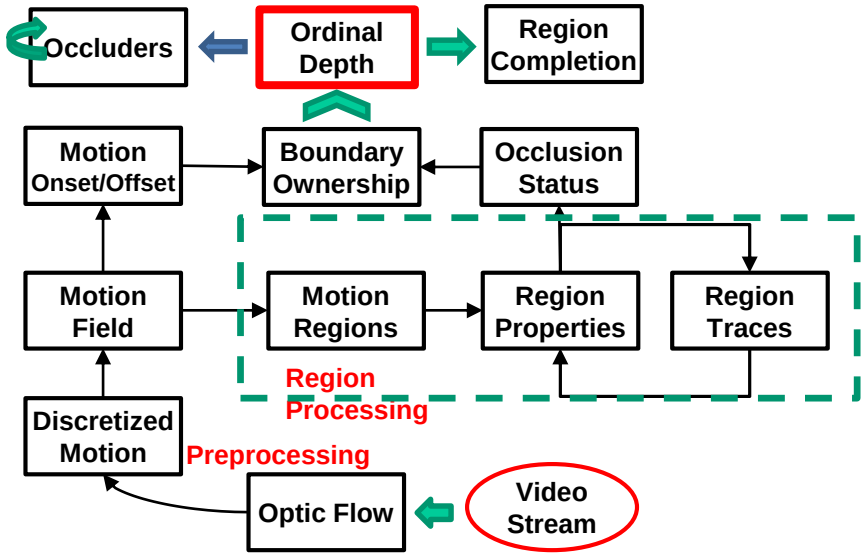
Northeastern University

Visualizing ordinal depth and occlusions.
Unoccluded moving parts of an object are highlighted.
Occluder is marked in red.

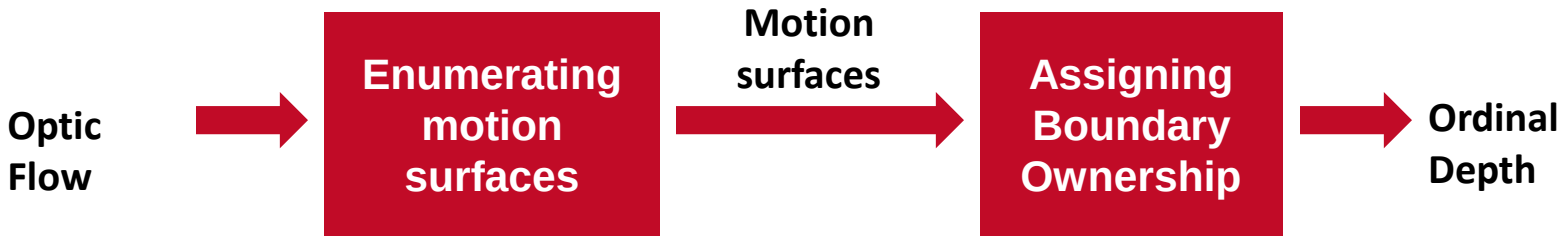
Goal: "direct" readout of "in front of", "behind", "emerging", or "disappearing" in video streams

Scene segmentation
- Based on *motion signals only*
  - Not contrast edges, stereo, ...

Use CogX, software from HPE Labs
- Maximize use of GPUs
- Near real-time processing, ~2 fps on an *HP Z820* workstation
  - Some processing in CPU kernels



CogMO – Ordinal Depth



Functional Control Flow of CogMO Algorithm

Video: CogMO algorithm

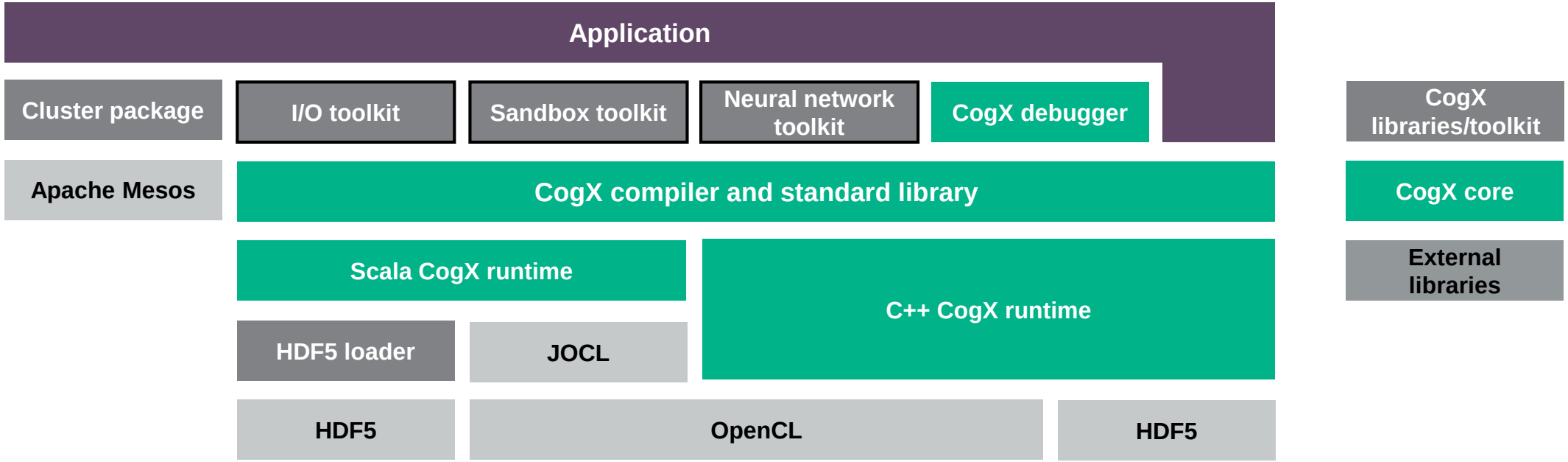
Roadmap

–Motivating evidence

–The CogX project and vision

–**Open-source availability**

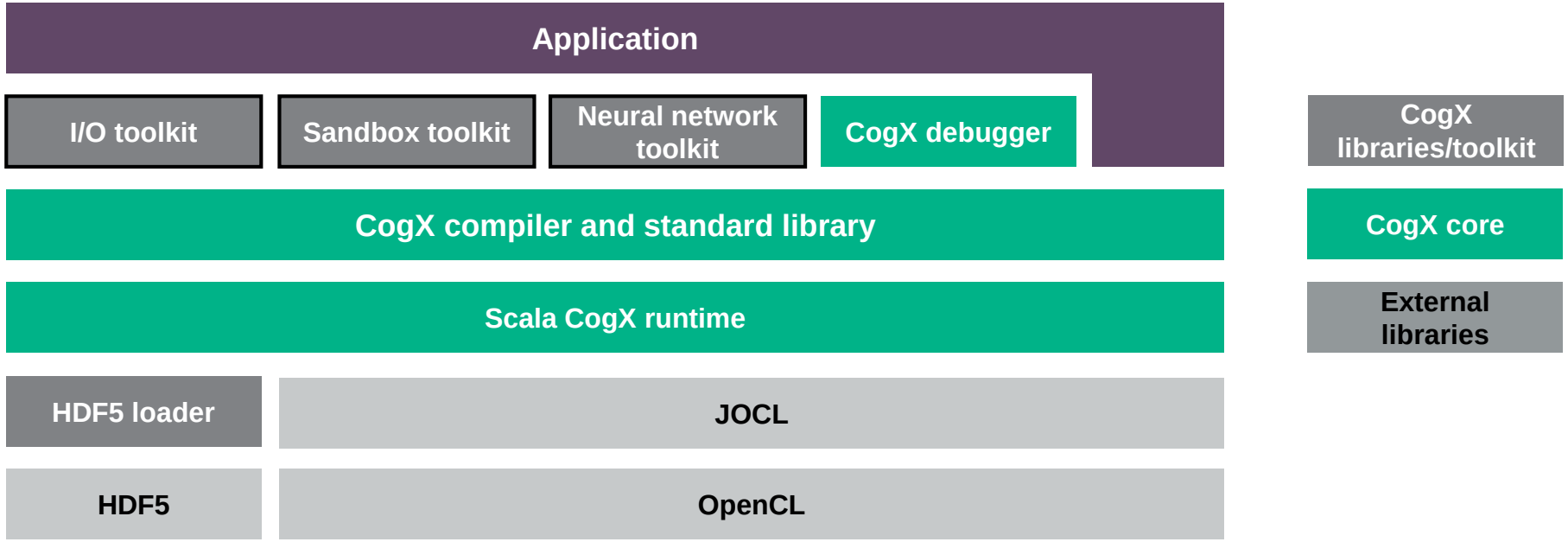
HPE Cognitive Computing Toolkit



Applications are written by users

- Introductory and training examples for single-GPU and distributed computation
- Performance benchmarks covering the core and neural network package
- Several larger-scale demo applications integrating multiple CogX functions

HPE Cognitive Computing Toolkit



Applications are written by users

- Introductory and training examples for single-GPU and distributed computation
- Performance benchmarks covering the core and neural network package
- Several larger-scale demo applications integrating multiple CogX functions

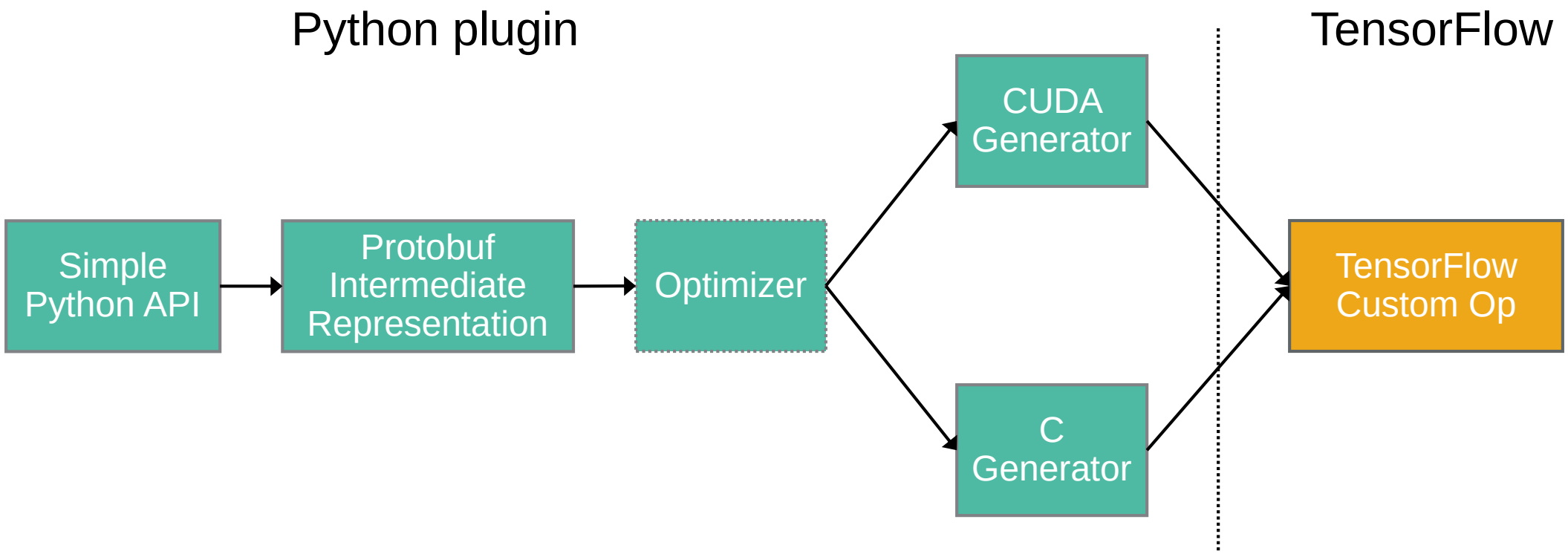
High-level comparison

	CogX	**TensorFlow**
Core data abstraction	Tensor Fields: single precision, restriction on dimensions	Tensors: typed multi-dimensional array
Core compute abstraction	OpenCL functions emitted and compiled at runtime. User kernels.	C++/CUDA functions compiled into TensorFlow project
Graph optimizations	Kernel fusion	Not available
Distribution across GPUs	Simulated annealing placer	Unreleased: Graph partitioning, Greedy placer
Debugging	Single-step runtime debugging. Text based profiler.	Non-interactive log file parser. Better graph visualization. Unreleased profiler.
Automatic differentiation	Supported as a library for neural network specific operations	Supported by most of core API
Fault tolerance	Not yet implemented	Automatic check-pointing and restart of graph
Control flow	Not yet implemented	Predicated execution
Runtime optimization	Not yet implemented	Interleaved processing of iterations, placer

High-level comparison

	CogX	TensorFlow
Core data abstraction	Tensor Fields: single precision, restriction on dimensions	Tensors: typed multi-dimensional array
Core compute abstraction	OpenCL functions emitted and compiled at runtime. User kernels.	C++/CUDA functions compiled into TensorFlow project
Graph optimizations	Kernel fusion	Not available
Distribution across GPUs	Simulated annealing placer	Unreleased: Graph partitioning, Greedy placer
Debugging	Single-step runtime debugging. Text based profiler.	Non-interactive log file parser. Better graph visualization. Unreleased profiler.
Automatic differentiation	Supported as a library for neural network specific operations	Supported by most of core API
Fault tolerance	Not yet implemented	Automatic check-pointing and restart of graph
Control flow	Not yet implemented	Predicated execution
Runtime optimization	Not yet implemented	Interleaved processing of iterations, placer

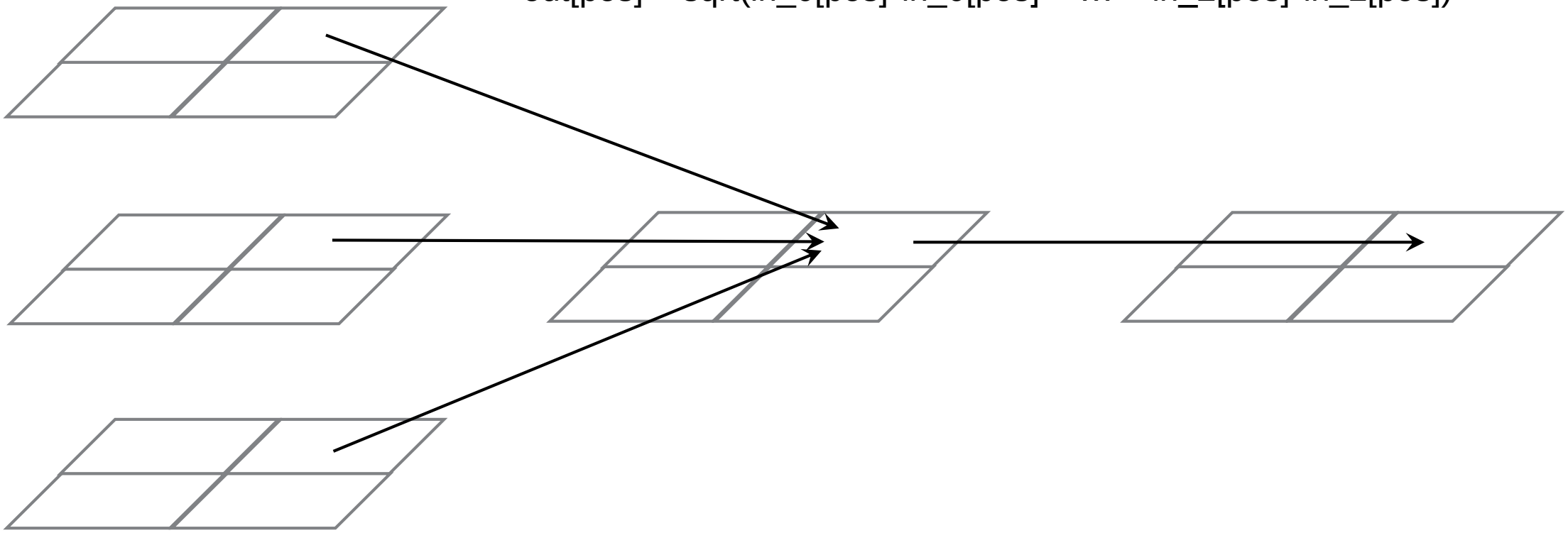
TensorFlow plugin: high productivity, high performance operators



TensorFlow plugin: a familiar programming model

Example: element-wise L2 Norm of three 2 x 2 tensors

out[pos] = sqrt(in_0[pos]*in_0[pos] + … + in_2[pos]*in_2[pos])



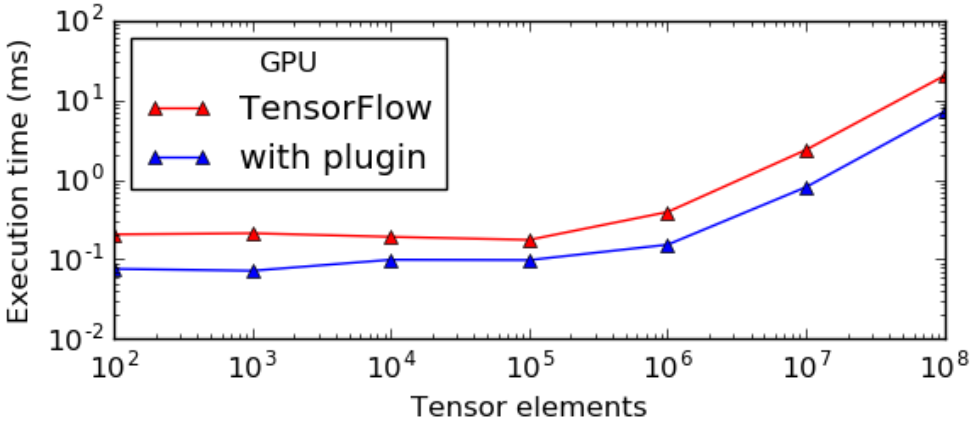
Input tensors **Workgroup shape** **Output tensor**

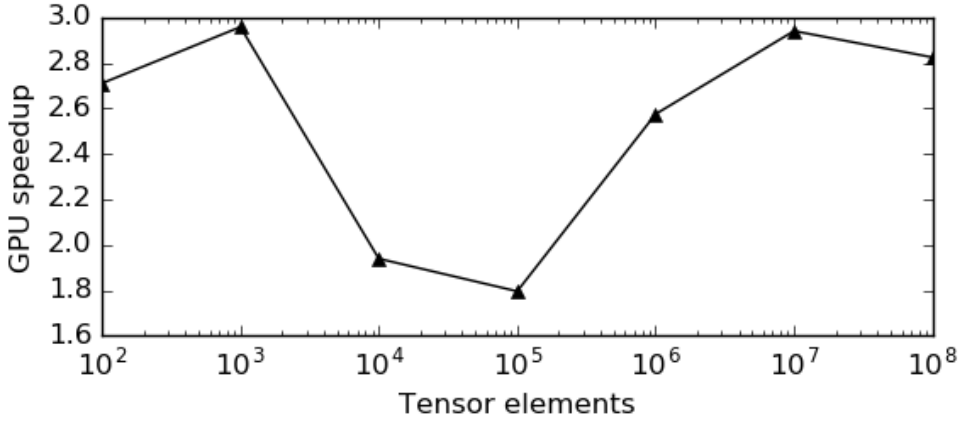
TensorFlow plugin: high productivity, high performance

High productivity:

```
def op(in0, in1, in2):
    pos = position_in(in0.shape)
    out = output_like(in0)

    a = in0[pos]
    b = in1[pos]
    c = in2[pos]

    out[pos] = sqrt(a*a + b*b + c*c)

    return out
```

High performance:





Hewlett Packard
Enterprise

Ben Chandler
benjamin.chandler@hpe.com