Embedding Foreign Code

Robert Clifton-Everest
University of New South Wales

Jointly with:
Trevor L. McDonell
Manuel M. T. Chakravarty
Gabriele Keller

Accelerate

```
dotp :: Acc (Vector Float) -> Acc (Vector Float) -> Acc (Scalar Float)
dotp xs ys = fold (+) 0 (zipWith (*) xs ys)


type Scalar e = Array Z e
type Vector e = Array (Z:.Int) e
type Matrix e = Array (Z:.Int:.Int) e
```

Skeletons

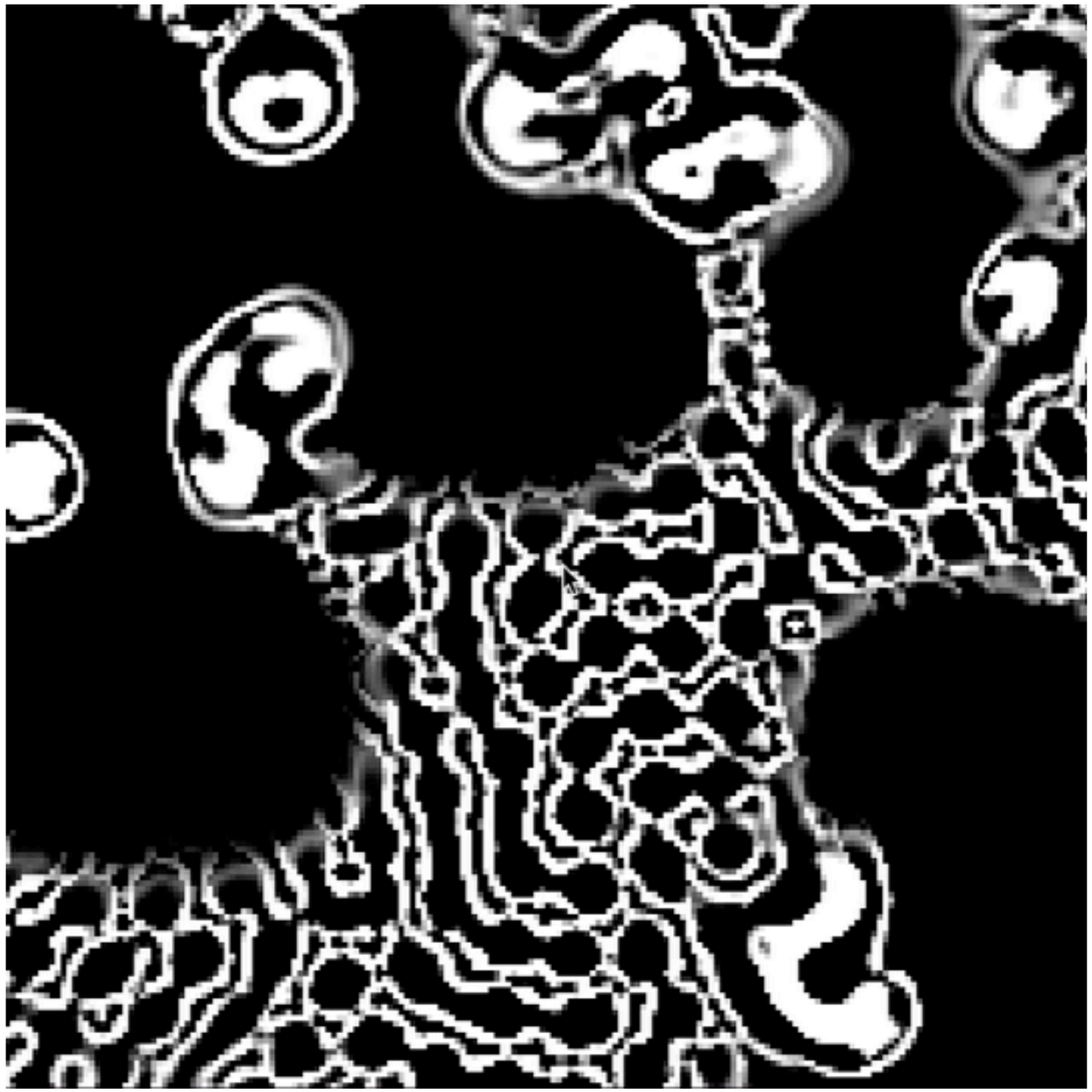
```
[cunit|
  __global__ void map( $params:argIn, $params:argOut )
  {
      const int shapeSize      = size(shOut);
      const int gridSize       = $exp:(gridSize dev);

      for (int ix = $exp:(threadIdx dev); ix < shapeSize; ix += gridSize)
      {
          $items:(dce x         .=. get ix)
          $items:(setOut "ix" .=. f x)
      }
  }
|]
```

```
data DelayedAcc a where
  Delayed :: (Shape sh, Elt e)
          => Exp sh                          -- array extent
          -> Fun (sh  -> e)                  -- generate element at index
          -> Fun (Int -> e)                  -- ...at linear index
          -> DelayedAcc (Array sh e)
```

```
[cunit|
  __global__ void foldAll( $params:argIn, $params:argOut )
  {   // omitted variable declarations
      if ( ix < shapeSize ) {
          $items:(y .=. get ix)

          for ( ix += gridSize; ix < shapeSize; ix += gridSize ) {
              $items:(x .=. get ix)
              $items:(y .=. combine x y)
          }
      }
      $items:(sdata "threadIdx.x" .=. y)
      __syncthreads();
      $stms:(treeReduce dev combine sdata)
      // first thread writes the result to memory
  }
|]
```

```
  const Int64 v2 = ix;
  const int v3 = toIndex(shIn0, shape(v2));
  const int v4 = toIndex(shIn1, shape(v2));
  float y0 = arrIn0_a0[v3] * arrIn1_a0[v4];
```

Foreign libraries



User's view

```
foreign import ccall "cublas_v2.h cublasSdot_v2" cublasSdot
  :: Handle
  -> Int                          -- Number of array elements
  -> DevicePtr Float -> Int       -- The two input arrays, and...
  -> DevicePtr Float -> Int       -- ...element stride
  -> DevicePtr Float              -- Result array
  -> IO ()
```

```
dotp_cublas :: Handle
            -> (Vector Float, Vector Float)
            -> CIO (Scalar Float)
dotp_cublas handle (xs, ys) = do
  let n  = arraySize (arrayShape xs)     -- number of input elements
  result    <- allocateArray Z           -- allocate a new Scalar array
  ((),xptr) <- devicePtrsOfArray xs      -- get device memory pointers
  ((),yptr) <- devicePtrsOfArray ys
  ((),rptr) <- devicePtrsOfArray result
  liftIO $ cublasSdot handle n xptr 1 yptr 1 rptr
  return result
```

Internally

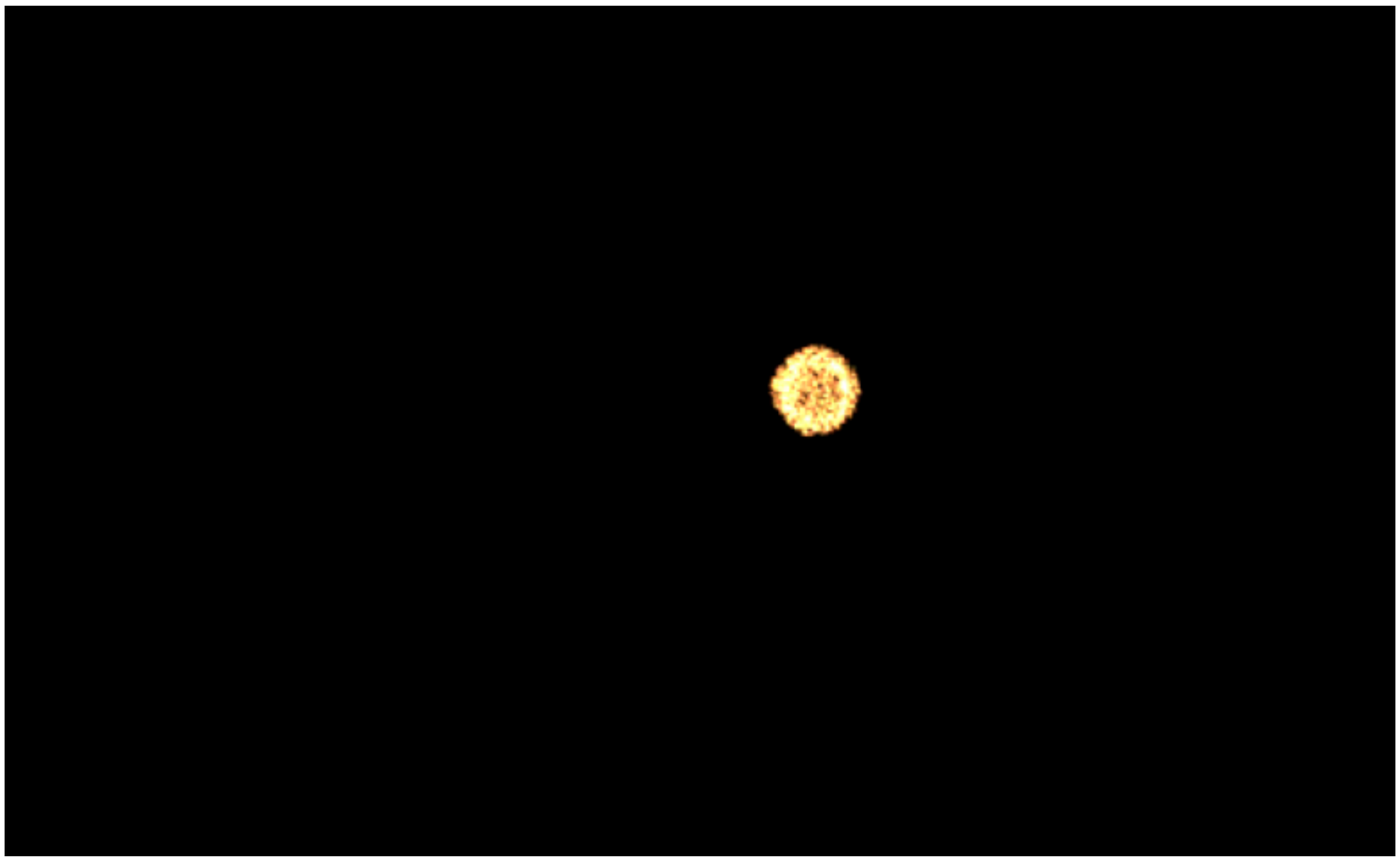
```
data Acc a where
  Map       :: (Elt a, Elt b, Shape sh)
            -> (Exp a -> Exp b)
            -> Acc (Array sh a)
            -> Acc (Array sh b)
...
Aforeign :: (Arrays as, Arrays bs, Foreign f)
         => f as bs                          -- foreign function
         -> (Acc as -> Acc bs)               -- fallback implementation
         -> Acc as                           -- input array
         -> Acc bs
```

```
class Typeable2 f => Foreign f where ...

data CUDAForeignAcc as bs = CUDAForeignAcc (as -> CIO bs)

instance Foreign CUDAForeignAcc where ...
```

Back to user

```
dotp' :: Acc (Vector Float) -> Acc (Vector Float) -> Acc (Scalar Float)
dotp' xs ys = Aforeign (CUDAForeignAcc (dotp_cublas handle))
                       (uncurry dotp)
                       (lift (xs, ys))
```

Accelerate as a C library



User's view

DotP.hs:

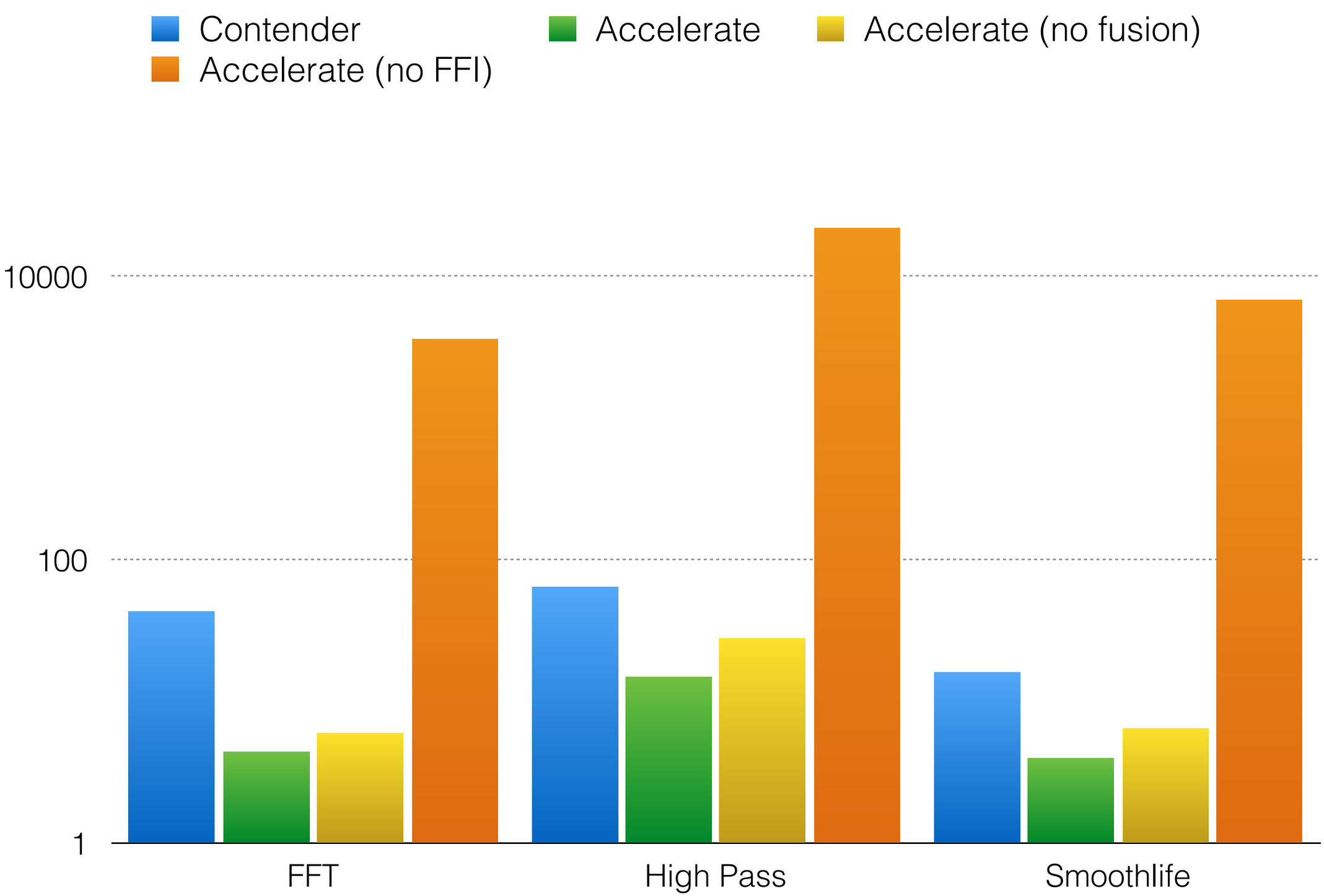
```
dotp :: Acc (Vector Float, Vector Float) -> Acc (Scalar Float)
dotp = uncurry $ \xs ys -> fold (+) 0 (zipWith (*) xs ys)

exportAfun 'dotp "dotp_compile"
```

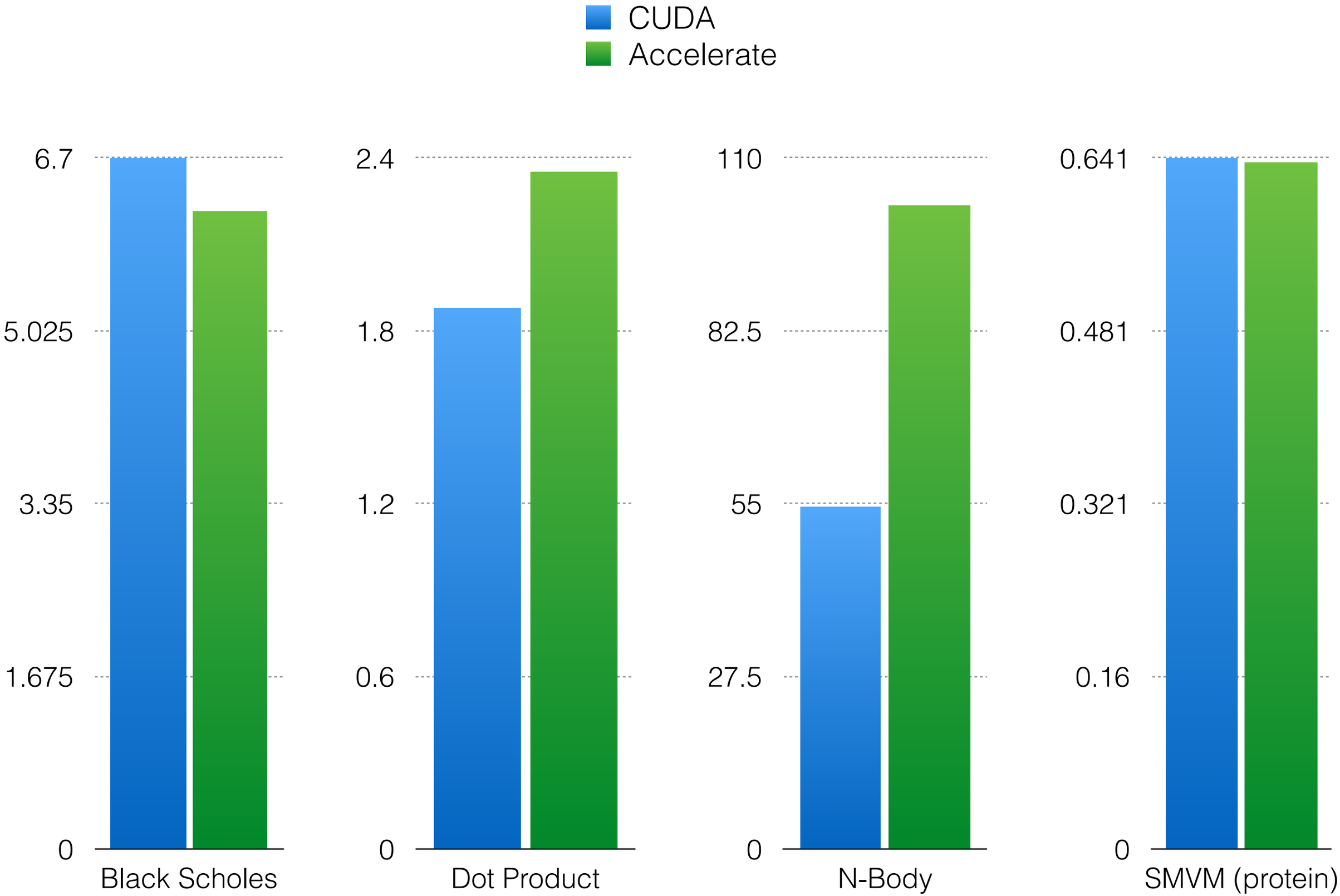
DotP_stub.h:

```
#include "HsFFI.h"
extern AccProgram dotp_compile(AccContext a1);
```

```c
#include "AccFFI.h"
#include "Dotp_stub.h"

OutputArray   out;
InputArray    in[2]   = { ... };
AccProgram    dotp    = dotp_compile( context );

runProgram( dotp, in, &out );
```

```c
typedef struct { int* shape; void** adata; } InputArray;

typedef struct { int* shape; void** adata;
                 HsStablePtr stable_ptr; } OutputArray;
```


Contender
Accelerate
Accelerate (no fusion)
Accelerate (no FFI)
10000
100
1
FFT
High Pass
Smoothlife


CUDA
Accelerate
6.7
5.025
3.35
1.675
0
Black Scholes
2.4
1.8
1.2
0.6
0
Dot Product
110
82.5
55
27.5
0
N-Body
0.641
0.481
0.321
0.16
0
SMVM (protein)

Questions?