Using Thrust to Sort CUDA FORTRAN Arrays

©NVIDIA Corporation | Summer, 2011 | Ty McKercher
CUDA How-To Guide

Value Proposition

- Implement high performance parallel applications with minimal programming effort
- Rich collection of object-based data parallel primitives to implement complex algorithms with concise, readable source code
- Now accessible via FORTRAN!

Acknowledgements

- Massimiliano Fatica (NVIDIA)
- Thomas Toy (Portland Group)
- Jared Hoberock (NVIDIA)
- Nathan Bell (NVIDIA)

Outline

- What is Thrust?
- Wrapper Functions
- Sort Implementation
- Perspectives

What is Thrust?

Standard template library for GPU

- Leverage Parallel Primitives for rapid development

Highly optimized algorithms

- Sorting, Prefix Sum, Reduction, more…

Ships free with CUDA v4.0

- Integrate with
  - CUDA C/C++
  - CUDA FORTRAN…

Thrust Components

Containers

- Manage host and device memory
- Simplify data transfers

Iterators

- Act like pointers
- Keep track of memory spaces

Algorithms

- Applied to Containers

Simple Sort Example using Thrust

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/generate.h>
#include <thrust/sort.h>
#include <thrust/copy.h>
#include <cstdlib>

int main(void)
{
        // generate 32M random numbers on the host
        thrust::host_vector<int> h_vec(32 << 20);
        thrust::generate(h_vec.begin(), h_vec.end(), rand);

        // transfer data to the device and sort
        thrust::device_vector<int> d_vec = h_vec;
        thrust::sort(d_vec.begin(), d_vec.end());

        // transfer data back to host
        thrust::copy(d_vec.begin(), d_vec.end(), h_vec.begin());
        return 0;
}
```

Outline

- What is Thrust?
- Wrapper Functions
- Sort Implementation
- Perspectives

Configure CUDA FORTRAN for CUDA v4.0

rc4.o

```
set CUDAROOT=/usr/local/cuda;
set CUDAVERSION=4.0;
```

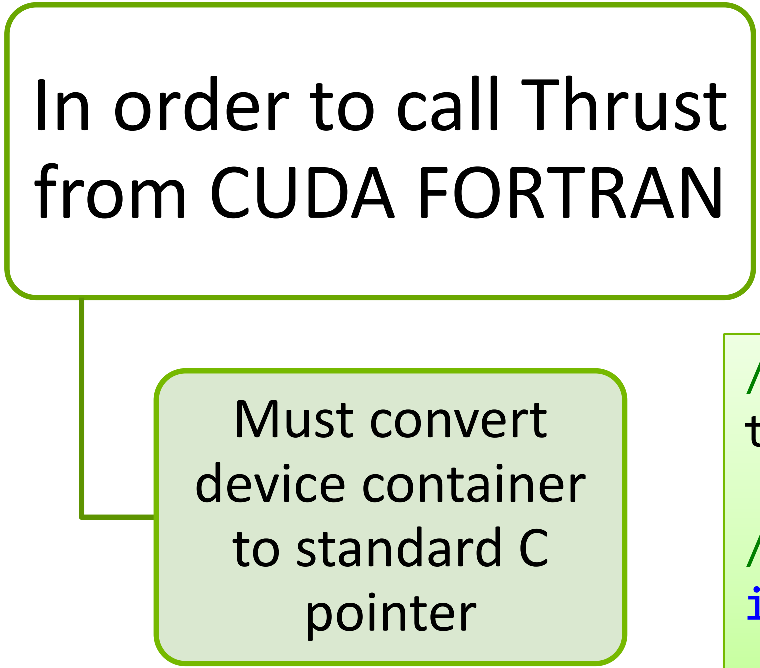
Makefile

```
NVINC = -I/usr/local/cuda/include
F90FLAGS = -rc=rc4.o -Mcuda=cc20 -O3
```

Until PGI supports CUDA v4.0 natively

- Create rc4.o file
- Compile CUDA FORTRAN files (.cuf) using –rc flag
- Add –L/usr/local/cuda/lib64 if using CUDA v4.0 Toolkit

Thrust conversion feature



In order to call Thrust from CUDA FORTRAN

Must convert device container to standard C pointer

```
// allocate device vector
thrust::device_vector d_vec(4);

// obtain raw pointer to device vector’s memory
int *ptr = thrust::raw_pointer_cast(&d_vec[0]);
```

Create C Wrapper to Thrust sort function

csort.cu

```
#include <thrust/device_vector.h>
#include <thrust/device_vector.h>
#include <thrust/sort.h>

extern "C" {
     void sort_int_wrapper( int *data, int N)
     {
          thrust::device_ptr<int> dev_ptr(data);
          thrust::sort(dev_ptr, dev_ptr+N);
     }

     void sort_float_wrapper( float *data, int N)
     {
          thrust::device_ptr<float> dev_ptr(data);
          thrust::sort(dev_ptr, dev_ptr+N);
     }

     void sort_double_wrapper( double *data, int N)
     {
          thrust::device_ptr<double> dev_ptr(data);
          thrust::sort(dev_ptr, dev_ptr+N);
     }
}
```

Makefile

```
NVINC = -I/usr/local/cuda/include
F90FLAGS = -rc=rc4.o -Mcuda=cc20 -O3

all: csort.o

csort.o: csort.cu
     nvcc -c -arch sm_13 $(NVINC) $^ -o $@

clean:
     rm csort.o
```

Add FORTRAN interface for Wrappers

thrust_module.cuf

```
module thrust
  interface thrustsort

    subroutine sort_int(input,N) bind(C,name="sort_int_wrapper")
    use iso_c_binding
       integer(c_int),device:: input(*)
       integer(c_int),value:: N
    end subroutine

    subroutine sort_float(input,N) bind(C,name="sort_float_wrapper")
    use iso_c_binding
      real(c_float),device:: input(*)
      integer(c_int),value:: N
    end subroutine

    subroutine sort_double(input,N) bind(C,name="sort_double_wrapper")
    use iso_c_binding
      real(c_double),device:: input(*)
      integer(c_int),value:: N
    end subroutine

  end interface
end module thrust
```

Makefile

```
NVINC = -I/usr/local/cuda/include
F90FLAGS = -rc=rc4.o -Mcuda=cc20 -O3

all: csort.o thrust_module.o

csort.o: csort.cu
     nvcc -c -arch sm_13 $(NVINC) $^ -o $@

thrust_module.o: thrust_module.cuf
     pgf90 –c $(F90FLAGS) $^ -o $@

clean:
     rm csort.o
```

Outline

- Using Thrust to sort FORTRAN array
- Wrapper Function
- Sort Implementation
- Perspectives

CUDA FORTRAN test sort program

test_sort.cuf

```fortran
program testsort
  use thrust

  real, allocatable :: cpuData(:)
  real, allocatable, device :: gpuData(:)
  integer:: N=10

  allocate(cpuData(N))
  allocate(gpuData(N))

  do i=1,N
    cpuData(i)=random(i)
  end do
  cpuData(5)=100.

  print *,"Before sorting", cpuData
  gpuData=cpuData
  call thrustsort(gpuData,size(gpuData))
  cpuData=gpuData
  print *,"After sorting", cpuData

end program
```

Makefile

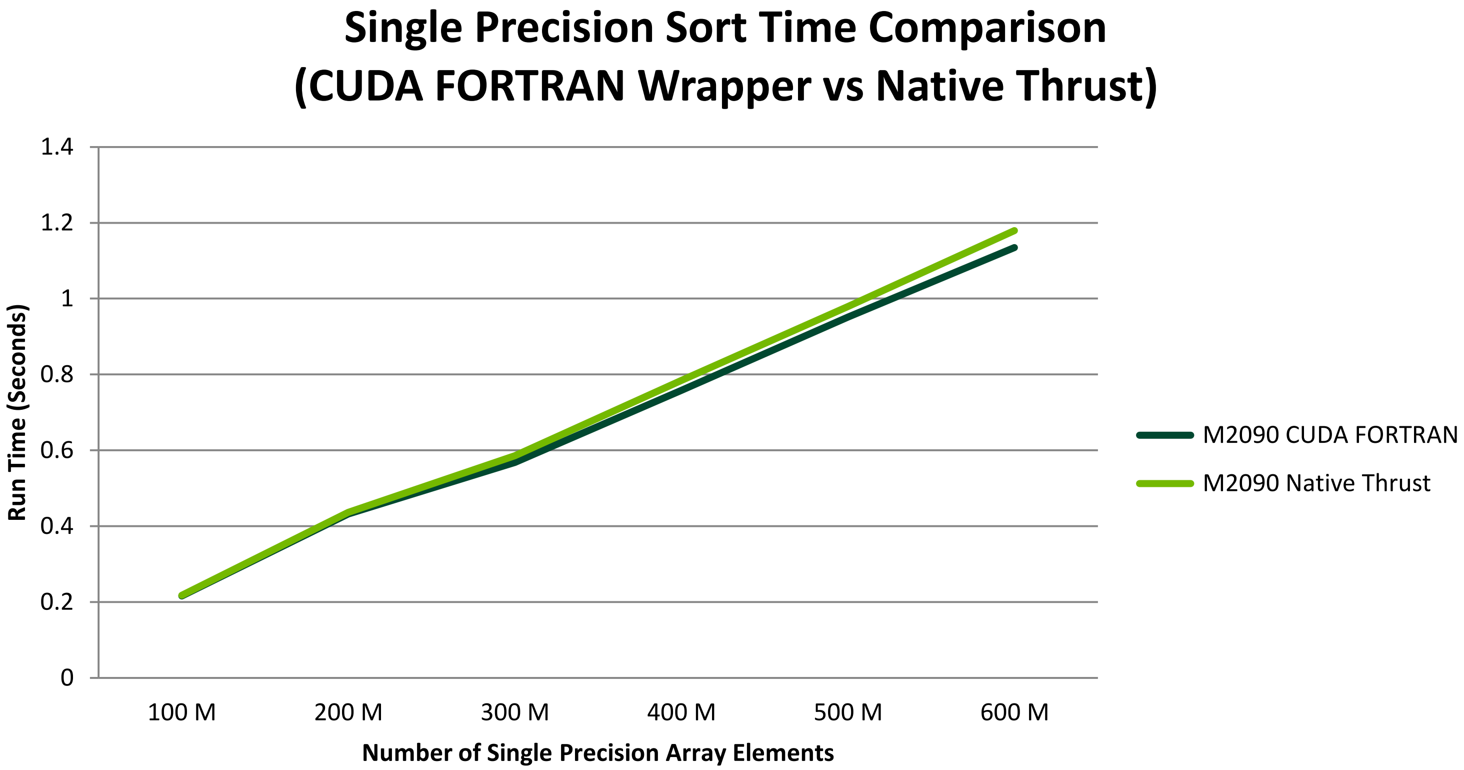
```makefile
NVINC = -I/usr/local/cuda/include
F90FLAGS = -rc=rc4.o -Mcuda=cc20 -O3

all: test_sort

test_sort: test_sort.o csort.o thrust_module.o
	pgf90 $(F90FLAGS) -o $@ $^

test_sort.o:    test_sort.cuf thrust_module.o
	pgf90 –c $(F90FLAGS) $< -o $@

csort.o: csort.cu
	nvcc -c -arch sm_13 $(NVINC) $^ -o $@

thrust_module.o: thrust_module.cuf
	pgf90 –c $(F90FLAGS) $^ -o $@

clean:
	rm csort.o test_sort.o test_sort
```

Add timing to sort program

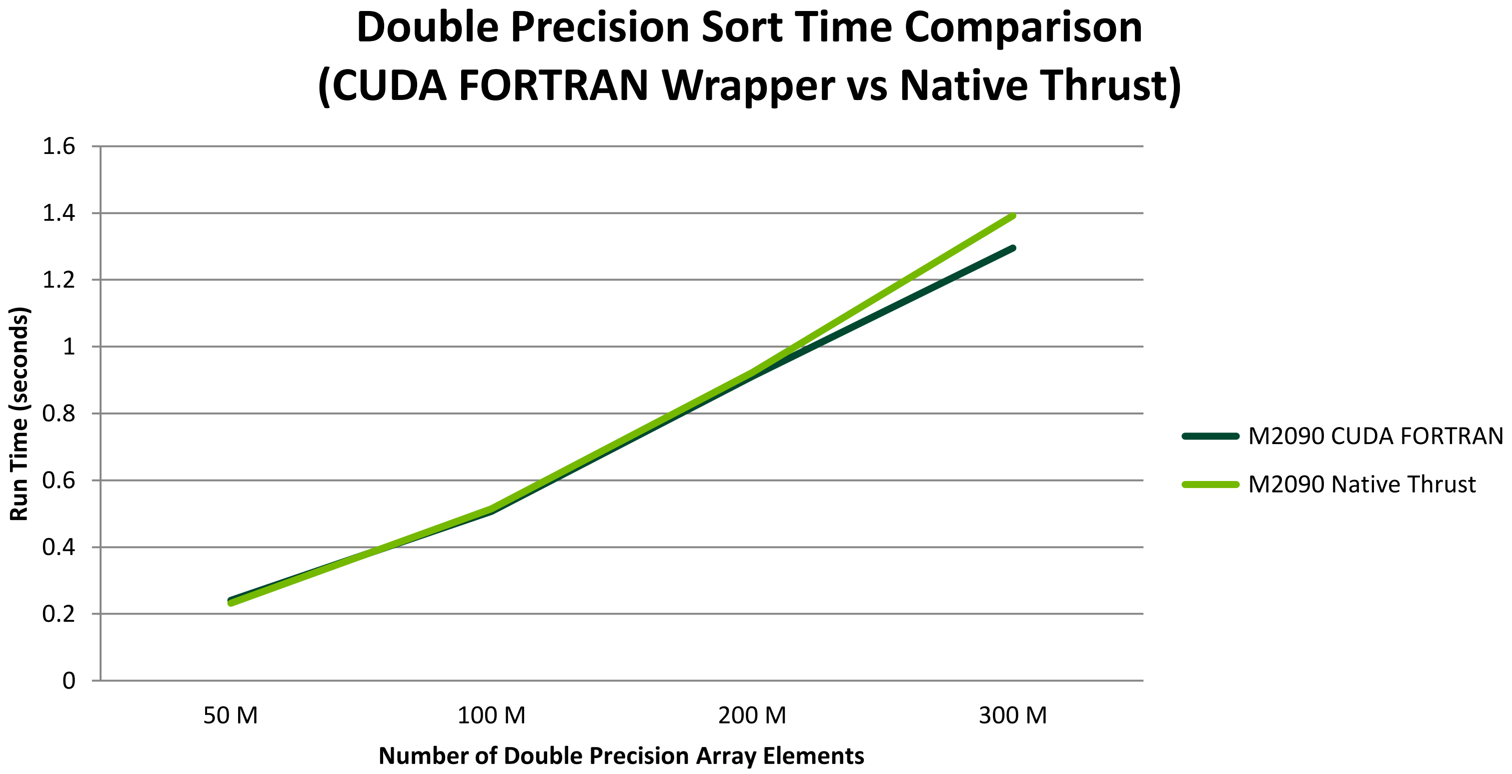
```
program timesort

  use cudafor
  use thrust
  implicit none

  real, allocatable :: cpuData(:)
  real, allocatable, device :: gpuData(:)
  integer:: i,N=100000000

  type ( cudaEvent ) :: startEvent , stopEvent
  real :: time, random
  integer :: istat

  istat = cudaEventCreate ( startEvent )
  istat = cudaEventCreate ( stopEvent )

  allocate(cpuData(N))
  allocate(gpuData(N))
```

```
  do i=1,N
    cpuData(i)=random(i)
  end do

  print *,"Sorting array of ",N, " single precision"

  gpuData=cpuData

  istat = cudaEventRecord ( startEvent , 0)
  call thrustsort(gpuData,size(gpuData))
  istat = cudaEventRecord ( stopEvent , 0)
  istat = cudaEventSynchronize ( stopEvent )
  istat = cudaEventElapsedTime ( time,startEvent,stopEvent )

  cpuData=gpuData

  print *," Sorted array in:",time," (ms)"
  print *,"After sort", cpuData(1:5),cpuData(N-4:N)

end program
```

Single Precision Timing Results



Double Precision Timing Results



Outline

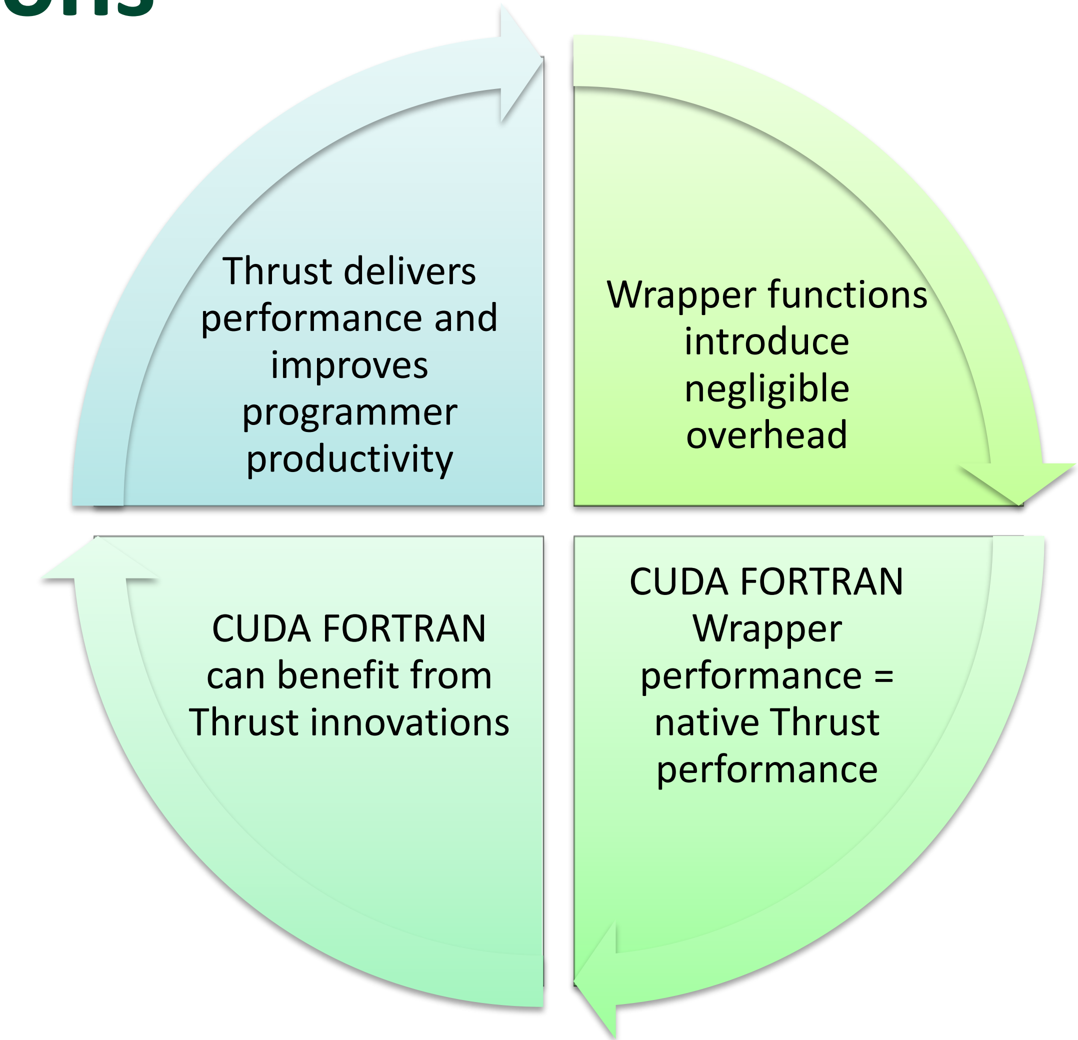
Using Thrust to sort FORTRAN array

Wrapper Functions

Sort Implementation

Perspectives

Conclusions



Thrust delivers performance and improves programmer productivity

Wrapper functions introduce negligible overhead

CUDA FORTRAN Wrapper performance = native Thrust performance

CUDA FORTRAN can benefit from Thrust innovations

References

- Collection of CUDA examples, tricks, and suggestions
  - http://cudamusing.blogspot.com/
- PGI Compiler and Tools
  - http://www.cse.scitech.ac.uk/events/GPU_2010/16_Miles.pdf
- Thrust Google Project Page
  - http://code.google.com/p/thrust/