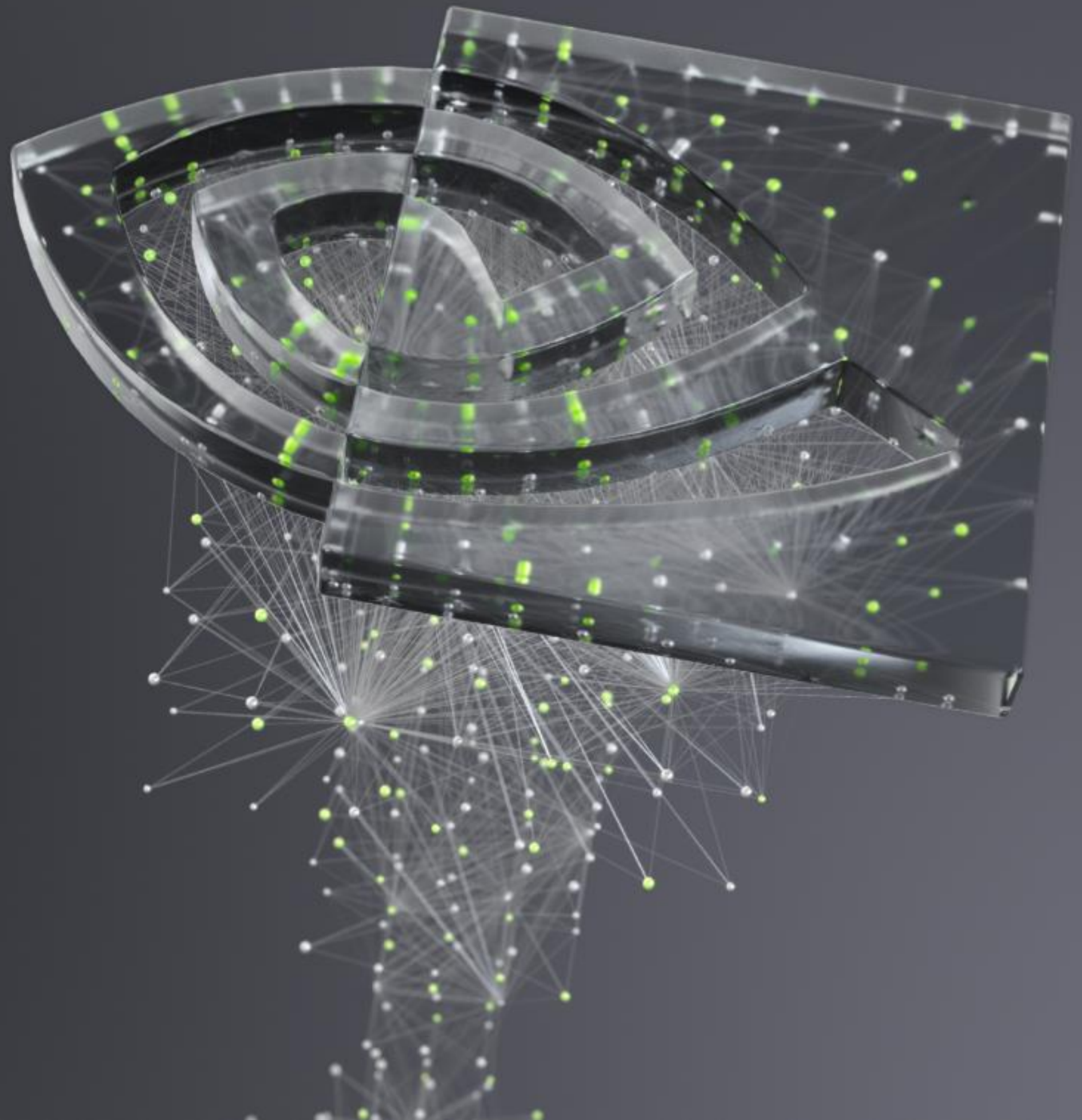
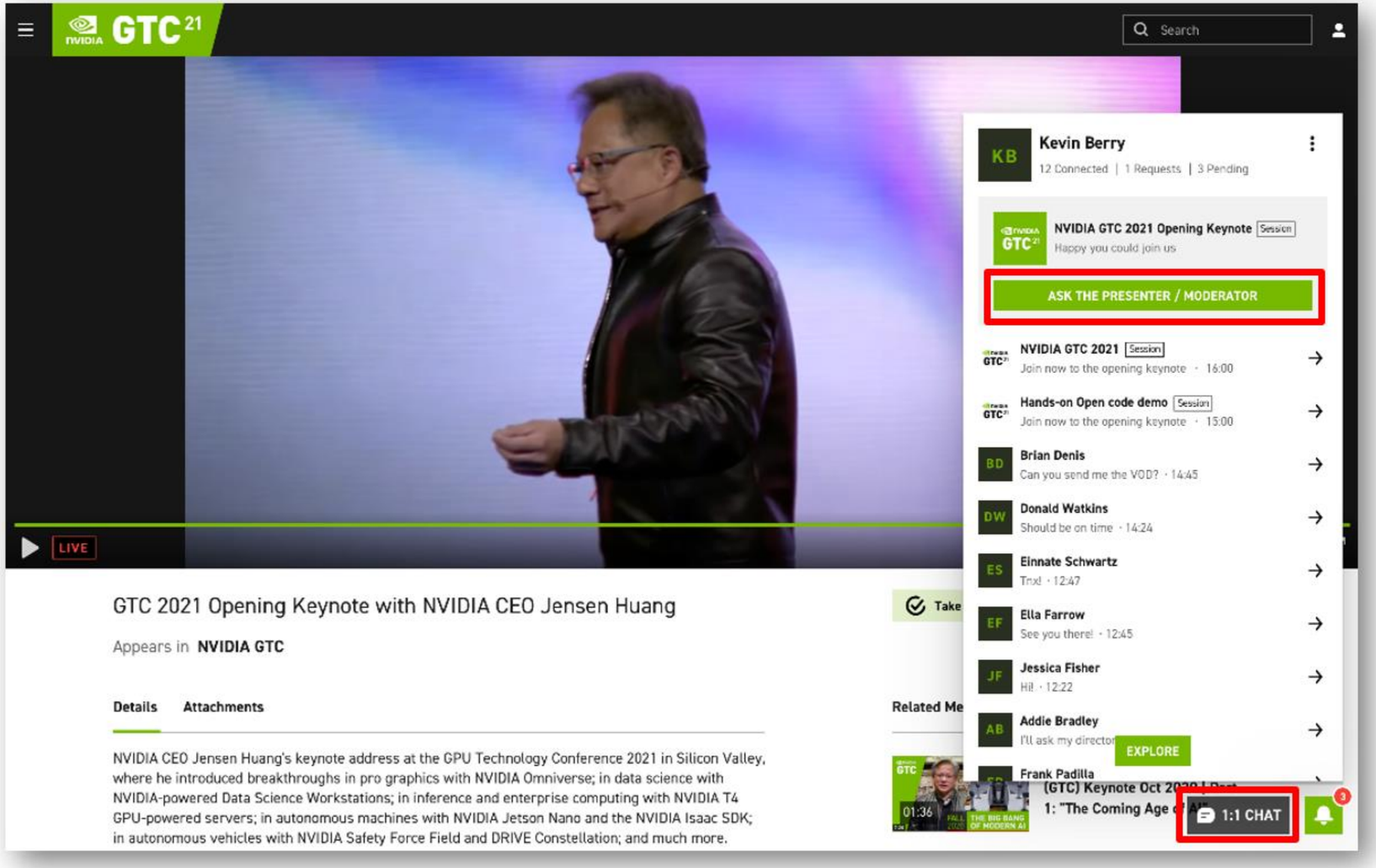
THRUST & THE C++ STANDARD ALGORITHMS

Conor Hoekstra, April 14, 2021

choekstra@nvidia.com / @code_report



I'M AVAILABLE TO CHAT DURING THIS SESSION



Click on "1:1 Chat," then "Ask the Presenter/Moderator" button to submit your question.
After the session is over, connect with me via attendee chat by searching for my name.

https://github.com/codereport/Talks

AGENDA

Introduction

- What is Thrust? What are the C++ standard algorithms?

The Difference

- set_difference(Thrust, C++ standard algorithms)

thrust::reduce vs std::accumulate

- How to parallelize reductions

Thrust Algorithms with Stencils

- A super useful overload

Fancy Iterators

- thrust::counting_iterator & friends

Algorithms + Data Structures = Programs

NVIDIA

RAPIDS

ode_report

YouTube

#include

- http://rapids.ai
- https://www.youtube.com/codereport
- https://www.adspthepodcast.com
- https://www.meetup.com/Programming-Languages-Toronto-Meetup/

Algorithms + Data Structures = Programs

INTRODUCTION

WHAT IS THRUST?

https://thrust.github.io/

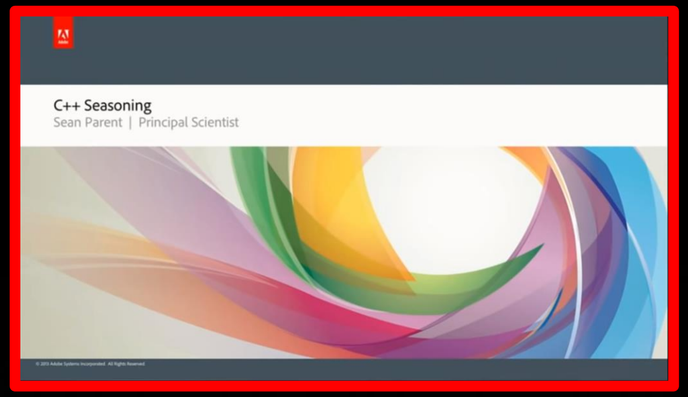
- Thrust is a **parallel algorithms** library which resembles the C++ standard algorithms
- Thrust's **high-level** interface greatly enhances programmer **productivity** while enabling performance portability between GPUs and multicore CPUs.
- **Interoperability** with established technologies (such as CUDA, TBB, and OpenMP) facilitates integration with existing software.
- Develop **high-performance** applications rapidly with Thrust!

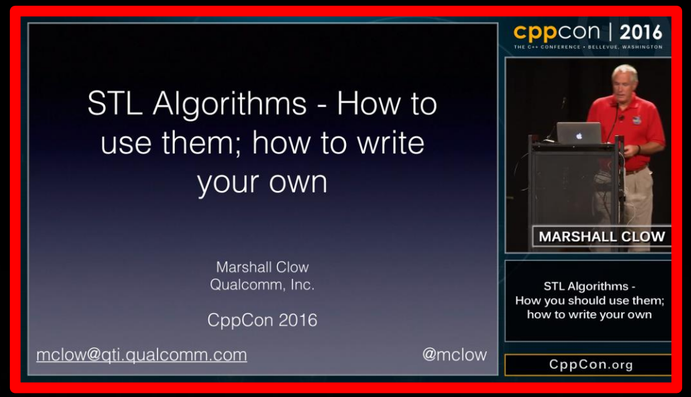
WHAT ARE THE C++ STANDARD ALGORITHMS?

https://en.cppreference.com/w/cpp/algorithm

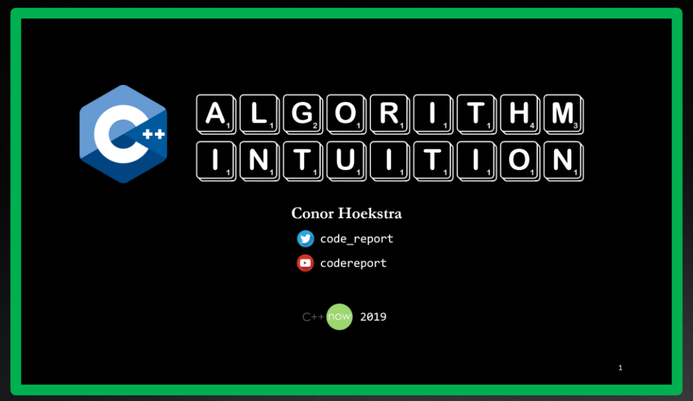
- The **algorithms library** defines functions for a variety of purposes (e.g. searching, sorting, counting, manipulating) that operate on ranges of elements.
- Algorithms can be found in:
  - **<algorithm>**
  - **<numeric>**
  - **<memory>**

RESOURCES


C++ Seasoning
Sean Parent | Principal Scientist


STL Algorithms - How to use them; how to write your own
Marshall Clow
Qualcomm, Inc.
CppCon 2016
mclow@qti.qualcomm.com
@mclow
cppcon | 2016
MARSHALL CLOW
STL Algorithms - How you should use them; how to write your own
CppCon.org

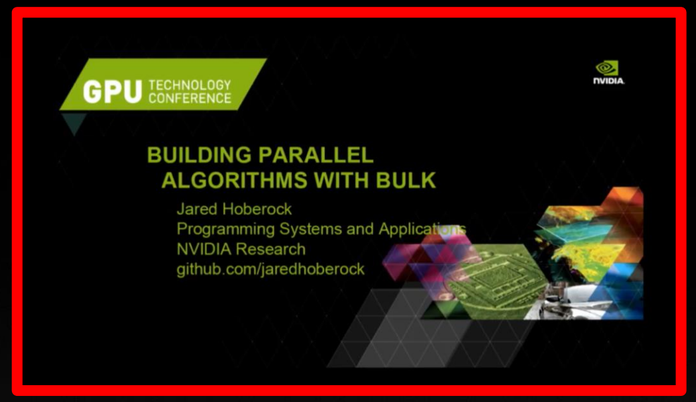

cppcon | 2018
105 STL ALGORITHMS IN LESS THAN AN HOUR
Agenda
Brief chat about STL algorithms
The STL algorithms themselves
@JoBoccara
JONATHAN BOCCARA
105 STL Algorithms in Less Than an Hour
CppCon.org

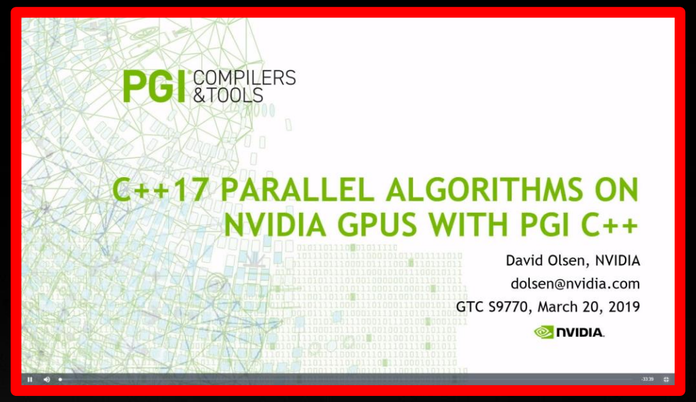

ALGORITHM INTUITION
Conor Hoekstra
code_report
codereport
C++now 2019
1


Better
Algorithm Intuition
Conor Hoekstra (he/him)
code_report
codereport
NVIDIA
RAPIDS
https://rapids.ai

RESOURCES


cppcon
the c++ conference
PARALLELIZING THE STANDARD
ALGORITHMS LIBRARY
Presenter: Jared Hoberock


GPU TECHNOLOGY CONFERENCE
NVIDIA
BUILDING PARALLEL
ALGORITHMS WITH BULK
Jared Hoberock
Programming Systems and Applications
NVIDIA Research
github.com/jaredhoberock


PGI COMPILERS &TOOLS
C++17 PARALLEL ALGORITHMS ON
NVIDIA GPUS WITH PGI C++
David Olsen, NVIDIA
dolsen@nvidia.com
GTC S9770, March 20, 2019
NVIDIA

https://github.com/codereport/Talks

THE DIFFERENCE

```
thrust::gather()
thrust::gather_if()
thrust::scatter()
thrust::scatter_if()
thrust::sequence()
thrust::stable_partition_copy()
thrust::tabulate()
thrust::transform_if()
// 12 *_by_key algorithms
```

```
thrust::sequence()
```

```
thrust::sequence()
std::iota()
```

```
thrust::exclusive_scan_by_key()
thrust::inclusive_scan_by_key()
thrust::merge_by_key()
thrust::reduce_by_key()
thrust::set_difference_by_key()
thrust::set_intersection_by_key()
thrust::set_symmetric_difference_by_key()
thrust::set_union_by_key()
thrust::sort_by_key()
thrust::stable_sort_by_key()
thrust::unique_by_key()
thrust::unique_by_key_copy()
```

Three problems…

```
thrust::gather()
thrust::gather_if()
thrust::scatter()
thrust::scatter_if()
thrust::sequence()
thrust::stable_partition_copy()
thrust::tabulate()
thrust::transform_if()
```

```
thrust::exclusive_scan_by_key()
thrust::inclusive_scan_by_key()
thrust::merge_by_key()
thrust::reduce_by_key()
thrust::set_difference_by_key()
thrust::set_intersection_by_key()
thrust::set_symmetric_difference_by_key()
thrust::set_union_by_key()
thrust::sort_by_key()
thrust::stable_sort_by_key()
thrust::unique_by_key()
thrust::unique_by_key_copy()
```

What is

`std::iota` + `std::transform` ?

What is

`thrust:sequence` + `thrust::transform` ?

```
thrust::tabulate()
```

First 10 odd numbers

Problem #1

```
auto odds = vector<int>(10);

iota(odds.begin(), odds.end(), 0);
transform(
    odds.begin(),
    odds.end(),
    odds.begin(),
    [](auto e) { return e * 2 + 1; });
```

Problem #1

```
auto odds = vector<int>(10);

thrust::tabulate(
    odds.begin(), odds.end(),
    [] (auto e) { return e * 2 + 1; });
```

Thrust algorithms can run on both **host** (cpu) and **device** (gpu)

```
auto odds = vector<int>(10);

thrust::tabulate(
    odds.begin(), odds.end(),
    [] (auto e) { return e * 2 + 1; });
```

```
auto odds = thrust::device_vector<int>(10);

thrust::tabulate(
    odds.begin(), odds.end(),
    [] __device__ (auto e) { return e * 2 + 1; });
```

Copy every other number

thrust::gather()

Problem #2

```
auto const deck        = vector<int>{13, 2, 14, 3, 6, 7};
auto const gather_map = vector<int>{0, 2, 4};
auto       hand        = vector<int>(3);

// deals every second card to hand
thrust::gather(
    gather_map.cbegin(),
    gather_map.cend(),
    deck.cbegin(),
    hand.begin());
```

Problem #2

```
auto const deck        = vector<int>{13, 2, 14, 3, 6, 7};
auto const gather_map = vector<int>{0, 2, 4};
auto       hand        = vector<int>(3);

// deals every second card to hand
thrust::gather(
    gather_map.cbegin(),
    gather_map.cend(),
    deck.cbegin(),
    hand.begin()); // 13, 14, 6
```

MCO

Maximum Consecutive Ones

Problem #3

1 1 1 0 0 1 0 1 1 1 1

Problem #3

1 1 1 0 0 1 0 1 1 1 1

```
thrust::reduce_by_key()
```

Problem #3

```
auto const ones = vector<int>{1, 1, 1, 0, 1, 1, 1, 1};
auto       sums = vector<int>(3);

thrust::reduce_by_key(
    ones.cbegin(),                      // keys   input
    ones.cend(),
    ones.cbegin(),                      // values input
    thrust::make_discard_iterator(), // keys   output
    sums.begin());                      // values output

auto const max = *thrust::max_element(
    sums.cbegin(), sums.cend());
```

(ﾟ/+/¨)ﾉ

```
thrust::tabulate()
thrust::gather()
thrust::reduce_by_key()
```

THRUST::REDUCE VS STD::ACCUMULATE

How to sum even numbers in a list?

```
auto const vals = vector<int>{42, 1729, 4104};

auto const sum_evens = accumulate(
    vals.cbegin(), vals.cend(), 0,
    [](auto acc, auto val) {
        return acc + (val % 2 == 0 ? val : 0);
    });
```

```
auto const vals = vector<int>{42, 1729, 4104};

auto const sum_evens = thrust::transform_reduce(
    vals.cbegin(), vals.cend(),
    [](auto val) { return val % 2 == 0 ? val : 0; },
    0,
    thrust::plus<int>{});
```

```
vals = [42, 1729, 4104]

sum_evens = vals
    |> Enum.filter(&Integer.is_even/1)
    |> Enum.sum
```

```
(~2|⊢)+.×⊢
```

THRUST ALGORITHMS WITH STENCILS

How to conditionally do things?

```
thrust::copy_if()
thrust::gather_if()
thrust::remove_if()
thrust::remove_copy_if()
thrust::replace_if()
thrust::replace_copy_if()
thrust::scatter_if()
thrust::transform_if()
```

```
auto const owners = vector<string>{"Ashwin", "Lesley", "Sarah"};
auto const pets   = vector<string>{"Dog",    "Cat",    "Dog"};

auto dog_owners = vector<string>{};
copy_if(
    owners.cbegin(),
    owners.cend(),
    back_inserter(dog_owners),
    [pet = pets.cbegin()](auto const& owner) mutable {
        return *pet++ == "Dog";
    });
```

```
auto const owners = vector<string>{"Ashwin", "Lesley", "Sarah"};
auto const pets   = vector<string>{"Dog",    "Cat",    "Dog"};

auto dog_owners = vector<string>{};
thrust::copy_if(
    owners.cbegin(),
    owners.cend(),
    pets.cbegin(), // stencil
    back_inserter(dog_owners),
    [](auto const& pet) { return pet == "Dog"; });
```

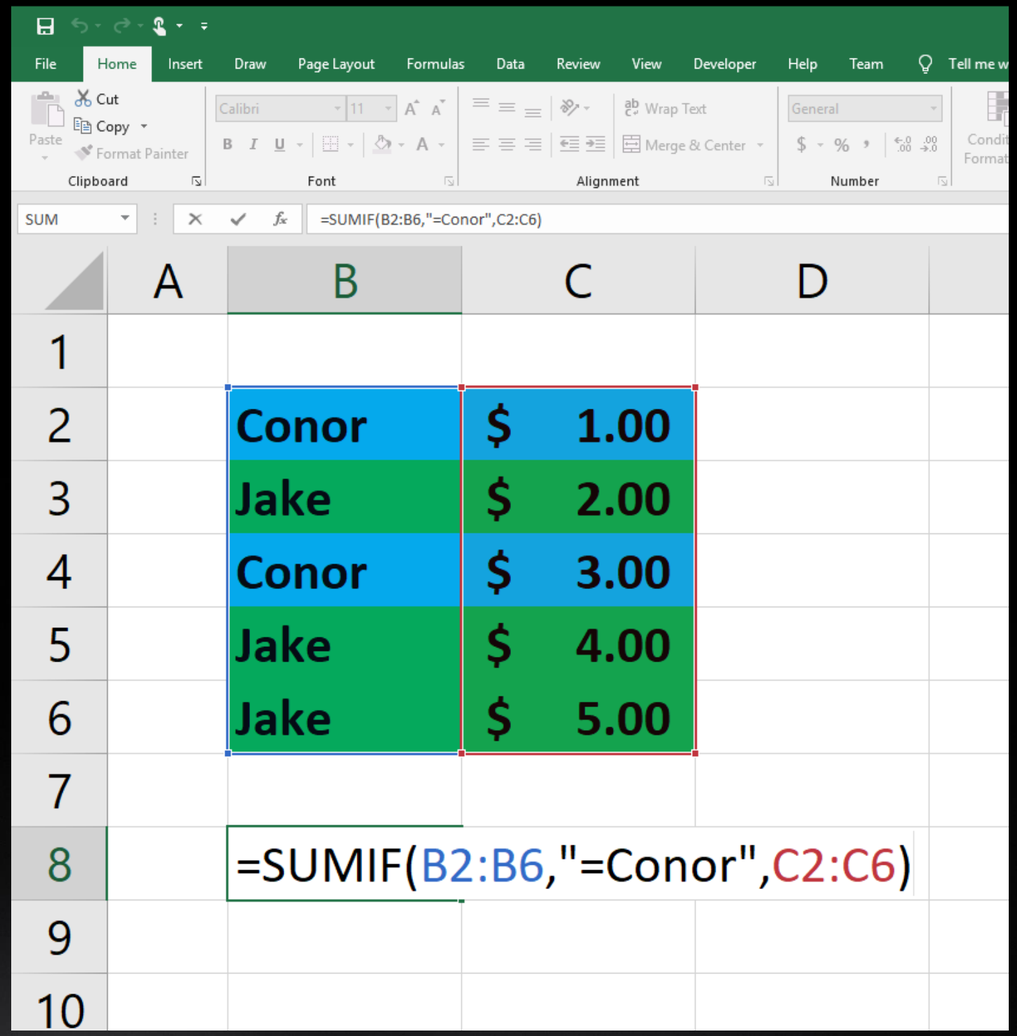
```
owners = ["Ahswin", "Lesley", "Sarah"]
pets   = ["Dog",    "Cat",    "Dog"  ]

dogOwners = owners
    |> Enum.zip(pets)
    |> Enum.filter(fn {_, pet} -> pet == "Dog" end)
    |> Enum.map(fn {owner, _} -> owner end)
```

```
owners ← 'Ashwin' 'Lesley' 'Sarah'
pets   ← 'Dog'    'Cat'    'Dog'

'Dog'∘≡¨pets
1 0 1
('Dog'∘≡¨pets)/owners
```

Ashwin	Sarah


File
Home
Insert
Draw
Page Layout
Formulas
Data
Review
View
Developer
Help
Team
Tell me w
Cut
Copy
Paste
Format Painter
Clipboard
Calibri
11
Font
Wrap Text
Merge & Center
Alignment
General
Number
SUM
=SUMIF(B2:B6,"=Conor",C2:C6)
A
B
C
D
1
2
3
4
5
6
7
8
9
10
Conor
$ 1.00
Jake
$ 2.00
Conor
$ 3.00
Jake
$ 4.00
Jake
$ 5.00
=SUMIF(B2:B6,"=Conor",C2:C6)

https://www.youtube.com/watch?v=C_lGkGwV4Xc

https://adspthepodcast.com/2021/02/19/Episode-13.html

```
thrust::copy_if()
thrust::gather_if()
thrust::remove_if()
thrust::remove_copy_if()
thrust::replace_if()
thrust::replace_copy_if()
thrust::partition()
thrust::scatter_if()
thrust::stable_partition()
thrust::transform()
thrust::transform_if()
```

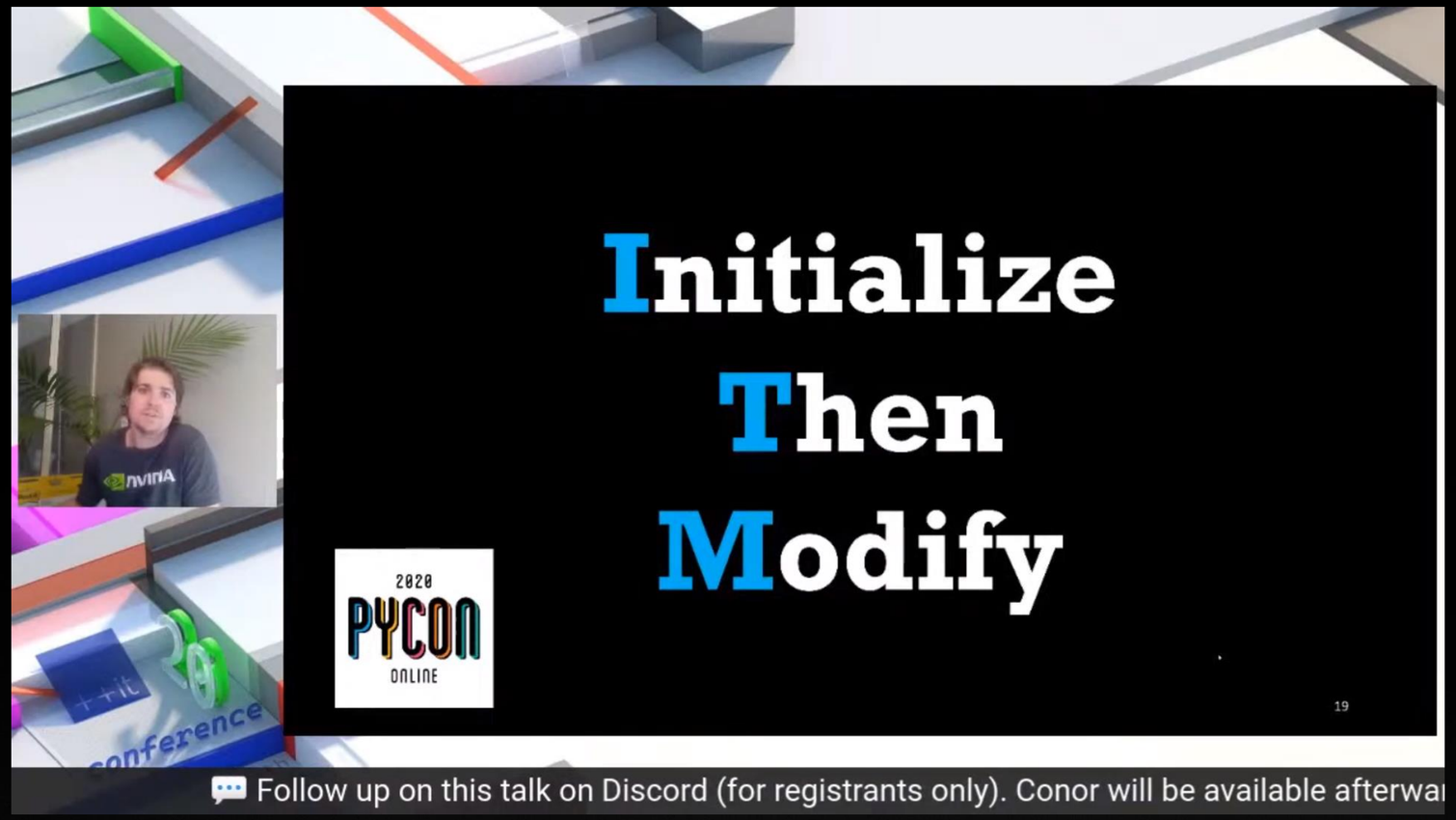
FANCY ITERATORS

```
auto ints = vector<int>(10);
thrust::sequence(ints.begin(), ints.end(), 0);
```

```
auto it = thrust::make_counting_iterator(0);
auto ints = vector<int>(it, it + 10);
```

```
auto it = thrust::make_counting_iterator(0);
auto const ints = vector<int>(it, it + 10);
```

https://youtu.be/CjHgL5EQdcY



https://youtu.be/CjHgL5EQdcY

```
auto it = thrust::make_counting_iterator(0);
auto const ints = vector<int>(it, it + 10);
```

```
thrust::constant_iterator()
thrust::counting_iterator()
thrust::discard_iterator()
thrust::permutation_iterator()
thrust::reverse_iterator()
thrust::transform_iterator()
thrust::transform_output_iterator()
thrust::zip_iterator()
```

```
    ⍳10
0 1 2 3 4 5 6 7 8 9
    2×⍳10
0 2 4 6 8 10 12 14 16 18
    1+2×⍳10
1 3 5 7 9 11 13 15 17 19
```

$$\underbrace{1+2\times}\,\underbrace{\iota\,10}$$

transform
transform_iterator

iota/sequence
counting_iterator

RAPIDS

counting_transform_iterator

```
auto odds = vector<int>(10);

thrust::tabulate(
    odds.begin(), odds.end(),
    [] (auto e) { return e * 2 + 1; });
```

Problem #1

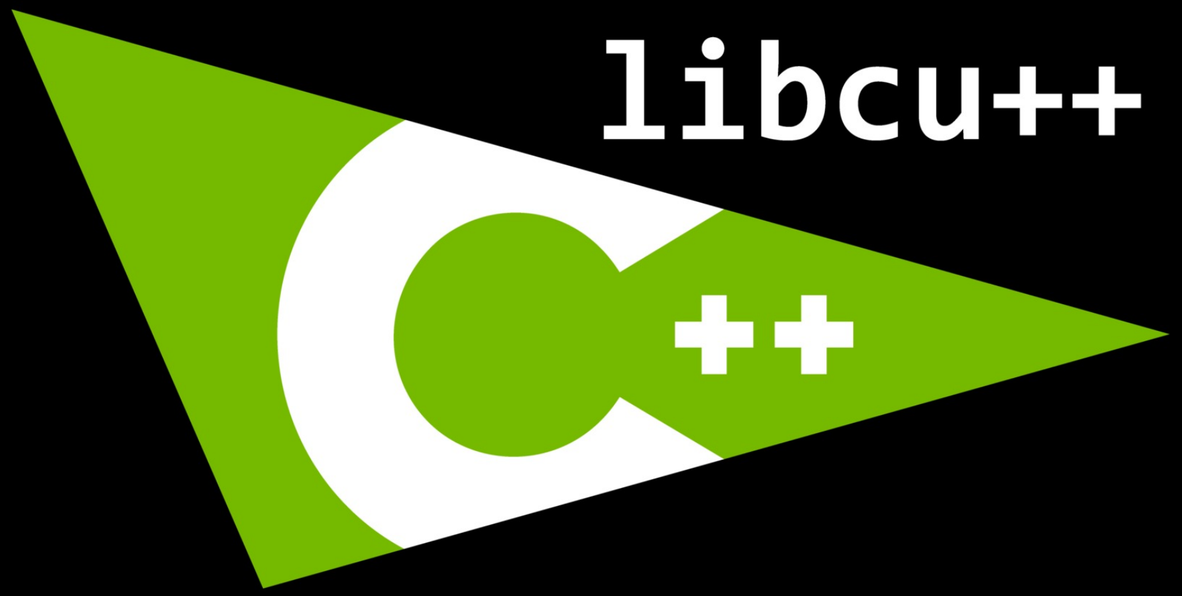
```
using namespace cudf::detail;
auto odd_fn = [](auto e) { return e * 2 + 1; };
auto it = make_counting_transform_iterator(0, odd_fn);
auto const odds = vector<int>(it, it + 10);
```

```
thrust::constant_iterator()
thrust::counting_iterator()
thrust::discard_iterator()
thrust::permutation_iterator()
thrust::reverse_iterator()
thrust::transform_iterator()
thrust::transform_output_iterator()
thrust::zip_iterator()
```

Can be used with C++ standard algorithms as well!

SUMMARY

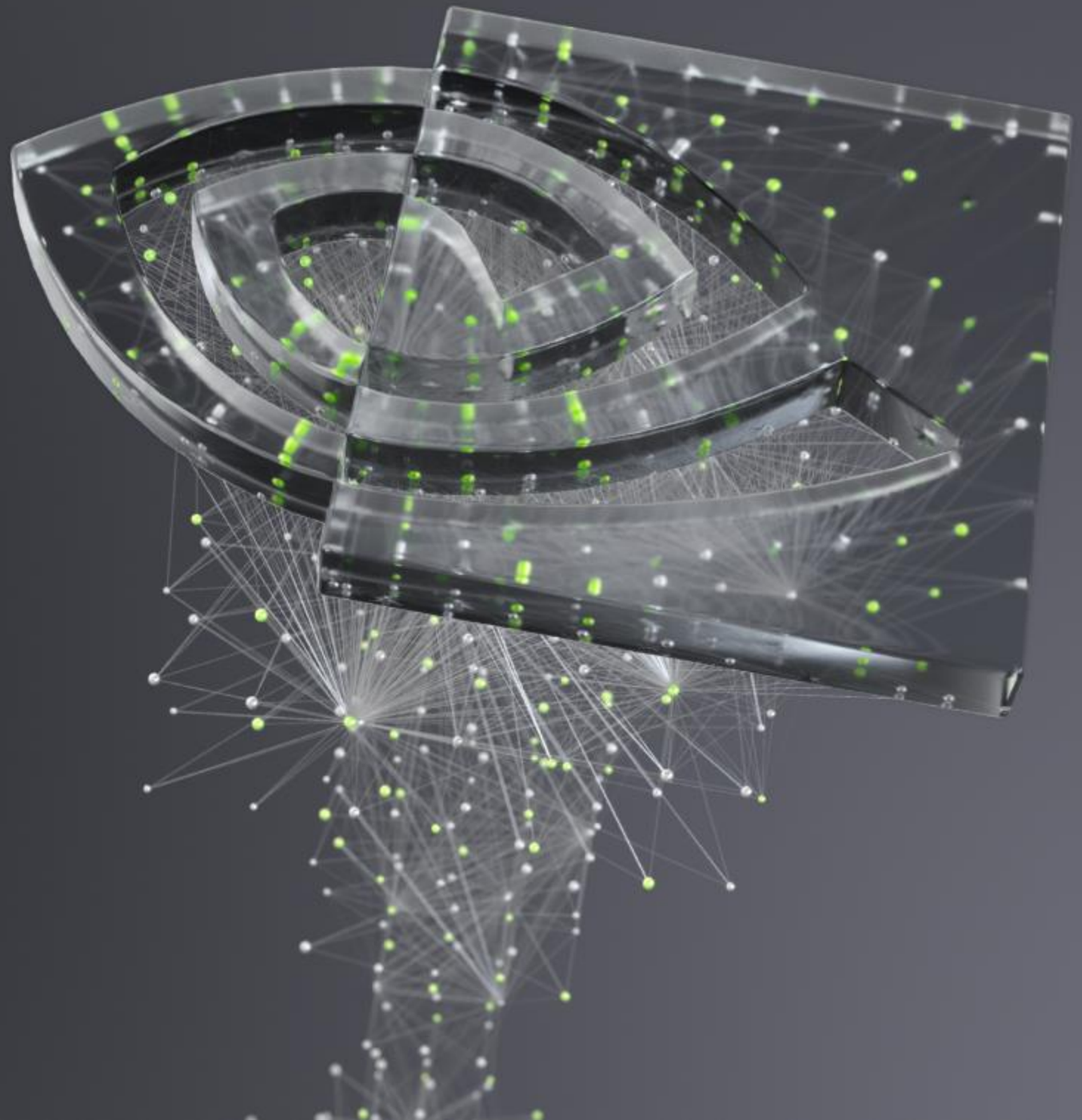
- **Thrust** has many algorithms not in C++ Standard Algorithms
- They run on both **device** and **host**
- **Stencil** overloads can be super useful
- **Fancy iterators** are awesome and can lead to more efficient and declarative code
- Be careful with algorithms like **thrust::reduce**
- **Parallel algorithms** have been added to C++17/20



The NVIDIA C++ Standard Library – Bryce Lelbach

https://thrust.github.io/

https://github.com/NVIDIA/thrust



QUESTIONS?

Conor Hoekstra, April 14, 2021

choekstra@nvidia.com / @code_report