

The logo for the GPU Technology Conference is located in the top-left corner. It consists of a green rectangular box with a small triangle pointing downwards on its left side. Inside the box, the text "GPU" is written in a large, bold, white sans-serif font, and "TECHNOLOGY CONFERENCE" is written in a smaller, white sans-serif font to its right.

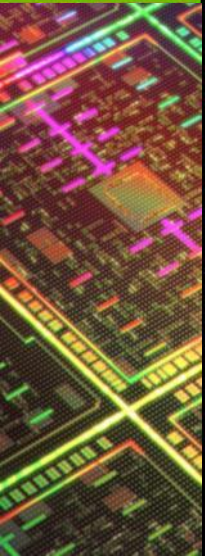
GPU TECHNOLOGY
CONFERENCE

The background of the slide is a close-up, top-down view of a GPU circuit board. The board is dark, and the intricate patterns of copper traces and components are highlighted with vibrant, multi-colored lights in shades of blue, green, yellow, orange, and red, creating a glowing, futuristic effect.

An Introduction to the Thrust Parallel Algorithms Library

What is Thrust?

- High-Level Parallel Algorithms Library
- Parallel Analog of the C++ Standard Template Library (STL)
- Performance-Portable Abstraction Layer
- Productive way to program CUDA



Example

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/sort.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
    thrust::device_vector<int> d_vec = h_vec;

    // sort data on the device
    thrust::sort(d_vec.begin(), d_vec.end());

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

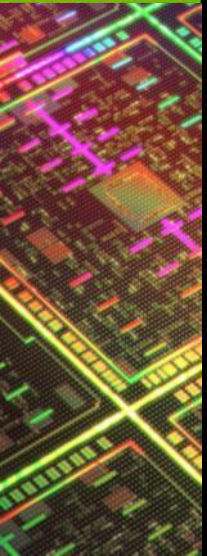
    return 0;
}
```

Easy to Use

- Distributed with CUDA Toolkit
- Header-only library
- Architecture agnostic
- Just compile and run!

```
$ nvcc -O2 -arch=sm_20 program.cu -o program
```

Why should I use Thrust?



Productivity

■ Containers

- `host_vector`
- `device_vector`

■ Memory Management

- Allocation
- Transfers

■ Algorithm Selection

- Location is implicit

```
// allocate host vector with two elements
thrust::host_vector<int> h_vec(2);

// copy host data to device memory
thrust::device_vector<int> d_vec = h_vec;

// write device values from the host
d_vec[0] = 27;
d_vec[1] = 13;

// read device values from the host
int sum = d_vec[0] + d_vec[1];

// invoke algorithm on device
thrust::sort(d_vec.begin(), d_vec.end());

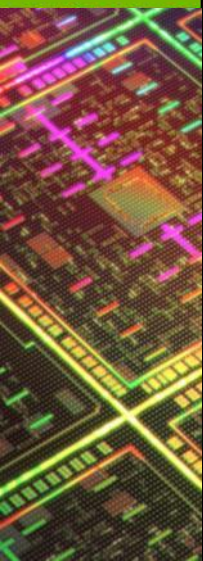
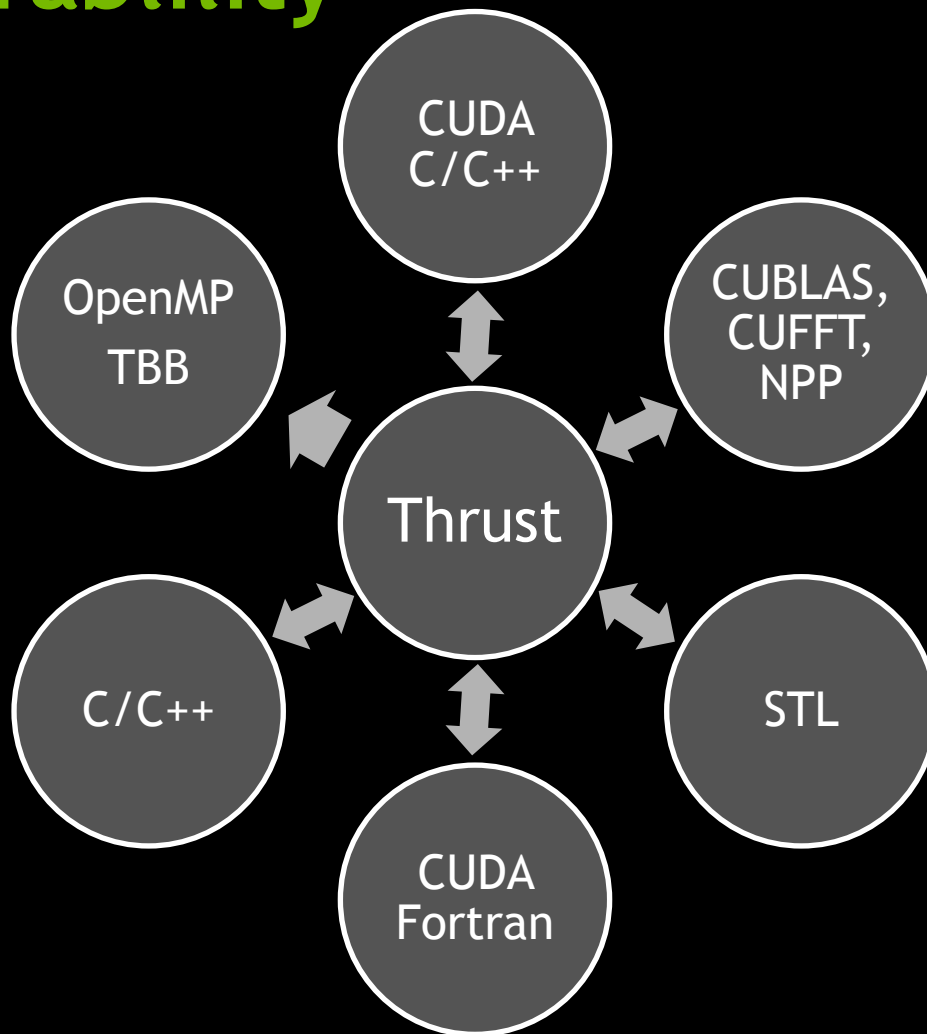
// memory automatically released
```

Productivity

- Large set of algorithms
 - ~75 functions
 - ~125 variations
- Flexible
 - User-defined types
 - User-defined operators

Algorithm	Description
<code>reduce</code>	Sum of a sequence
<code>find</code>	First position of a value in a sequence
<code>mismatch</code>	First position where two sequences differ
<code>inner_product</code>	Dot product of two sequences
<code>equal</code>	Whether two sequences are equal
<code>min_element</code>	Position of the smallest value
<code>count</code>	Number of instances of a value
<code>is_sorted</code>	Whether sequence is in sorted order
<code>transform_reduce</code>	Sum of transformed sequence

Interoperability



Portability

- Support for CUDA, TBB and OpenMP
 - Just recompile!

```
nvcc -DTHRUST_DEVICE_SYSTEM=THRUST_HOST_SYSTEM_OMP
```

NVIDIA GeForce GTX 580

```
$ time ./monte_carlo
pi is approximately 3.14159

real    0m6.190s
user    0m6.052s
sys     0m0.116s
```

Intel Core i7 2600K

```
$ time ./monte_carlo
pi is approximately 3.14159

real    1m26.217s
user    11m28.383s
sys     0m0.020s
```

Backend System Options

Host Systems

THRUST_HOST_SYSTEM_CPP
THRUST_HOST_SYSTEM_OMP
THRUST_HOST_SYSTEM_TBB

Device Systems

THRUST_DEVICE_SYSTEM_CUDA
THRUST_DEVICE_SYSTEM_OMP
THRUST_DEVICE_SYSTEM_TBB

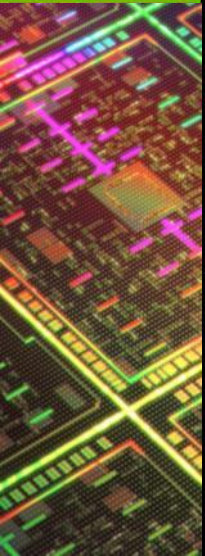
Multiple Backend Systems

- Mix different backends freely within the same app

```
thrust::omp::vector<float> my_omp_vec(100);  
thrust::cuda::vector<float> my_cuda_vec(100);  
  
...  
  
// reduce in parallel on the CPU  
thrust::reduce(my_omp_vec.begin(), my_omp_vec.end());  
  
// sort in parallel on the GPU  
thrust::sort(my_cuda_vec.begin(), my_cuda_vec.end());
```

Thrust_sort

- Open exercises/cuda/thrust_sort/kernel.cu
- Code should build without modification
- What is performance of GPU versus CPU code?



Potential Workflow

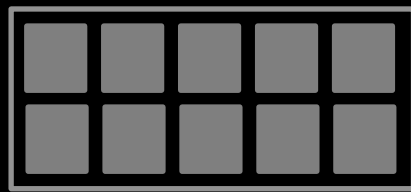
Thrust
Implementation



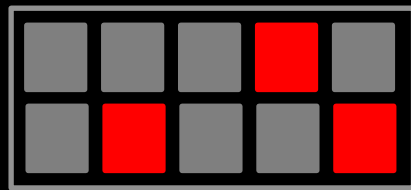
Profile
Application



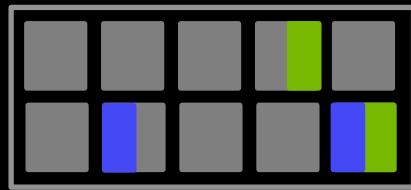
Specialize
Components



Application



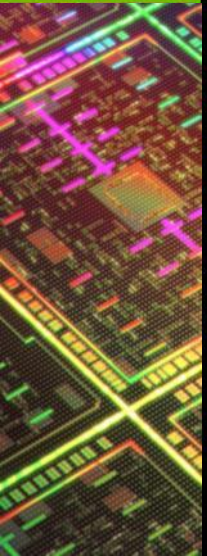
Bottleneck



Optimized Code

Robustness

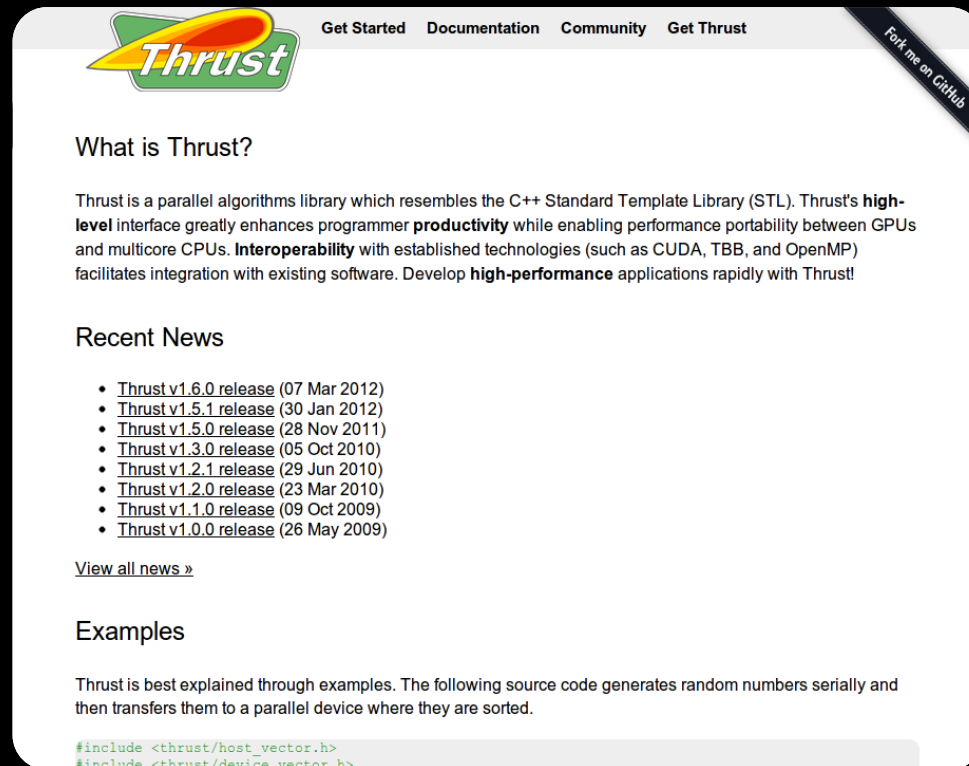
- Reliable
 - Supports all CUDA-capable GPUs
- Well-tested
 - ~850 unit tests run daily
- Robust
 - Handles many pathological use cases



Openness

- Open Source Software
 - Apache License
 - Hosted on GitHub
- Welcome to
 - Suggestions
 - Criticism
 - Bug Reports
 - Contributions

thrust.github.com



The screenshot shows the Thrust GitHub repository page. At the top, there is a navigation bar with links for "Get Started", "Documentation", "Community", and "Get Thrust". The Thrust logo is prominently displayed. A diagonal banner in the top right corner says "Fork me on GitHub". The main content area is titled "What is Thrust?" and contains a paragraph describing Thrust as a parallel algorithms library that resembles the C++ STL, highlighting its high-level interface, productivity, and interoperability with established technologies like CUDA, TBB, and OpenMP. Below this is a "Recent News" section with a list of release dates from 2009 to 2012. A link "View all news »" is provided. The "Examples" section begins with a paragraph explaining that Thrust is best understood through examples and is followed by a code block showing the inclusion of Thrust headers.

Get Started Documentation Community Get Thrust

Thrust

Fork me on GitHub

What is Thrust?

Thrust is a parallel algorithms library which resembles the C++ Standard Template Library (STL). 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!

Recent News

- [Thrust v1.6.0 release](#) (07 Mar 2012)
- [Thrust v1.5.1 release](#) (30 Jan 2012)
- [Thrust v1.5.0 release](#) (28 Nov 2011)
- [Thrust v1.3.0 release](#) (05 Oct 2010)
- [Thrust v1.2.1 release](#) (29 Jun 2010)
- [Thrust v1.2.0 release](#) (23 Mar 2010)
- [Thrust v1.1.0 release](#) (09 Oct 2009)
- [Thrust v1.0.0 release](#) (26 May 2009)

[View all news »](#)

Examples

Thrust is best explained through examples. The following source code generates random numbers serially and then transfers them to a parallel device where they are sorted.

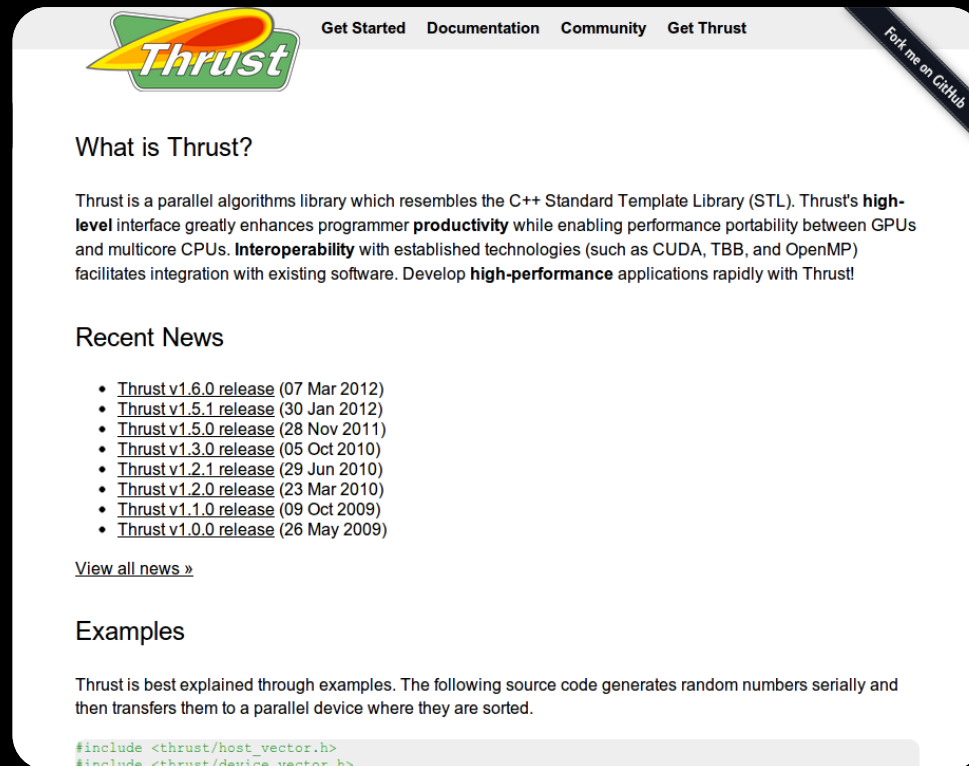
```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
```

Examples

Resources

- Documentation
- Examples
- Mailing List
- Webinars
- Publications

thrust.github.com



The screenshot shows the Thrust GitHub repository page. At the top, there is a navigation bar with links for "Get Started", "Documentation", "Community", and "Get Thrust". The Thrust logo is prominently displayed. A diagonal banner in the top right corner says "Fork me on GitHub". The main content area is titled "What is Thrust?" and contains a paragraph describing Thrust as a parallel algorithms library that resembles the C++ STL, highlighting its high-level interface, productivity, and interoperability with established technologies like CUDA, TBB, and OpenMP. Below this is a "Recent News" section with a list of release dates from 2009 to 2012. A link "View all news »" is provided. The "Examples" section begins with a paragraph explaining that Thrust is best understood through examples and provides a snippet of C++ code for generating random numbers.

Thrust

Get Started Documentation Community Get Thrust

Fork me on GitHub

What is Thrust?

Thrust is a parallel algorithms library which resembles the C++ Standard Template Library (STL). 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!

Recent News

- [Thrust v1.6.0 release](#) (07 Mar 2012)
- [Thrust v1.5.1 release](#) (30 Jan 2012)
- [Thrust v1.5.0 release](#) (28 Nov 2011)
- [Thrust v1.3.0 release](#) (05 Oct 2010)
- [Thrust v1.2.1 release](#) (29 Jun 2010)
- [Thrust v1.2.0 release](#) (23 Mar 2010)
- [Thrust v1.1.0 release](#) (09 Oct 2009)
- [Thrust v1.0.0 release](#) (26 May 2009)

[View all news »](#)

Examples

Thrust is best explained through examples. The following source code generates random numbers serially and then transfers them to a parallel device where they are sorted.

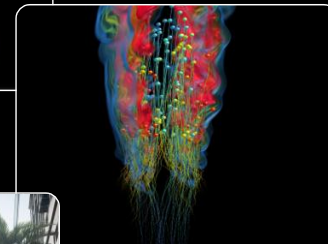
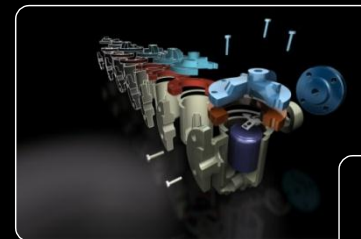
```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
```


GPU Technology Conference 2013

March 18-21 | San Jose, CA

Why attend GTC?

GTC advances global awareness of the dramatic changes we're seeing in science and research, graphics, cloud computing, game development, and mobile computing, and how the GPU is central to innovation in all areas.



Ways to participate

- Submit a Research Poster - share your work and gain exposure as a thought leader
- Register - learn from the experts and network with your peers
- Exhibit/Sponsor - promote your organization as a key player in the GPU ecosystem



Visit www.gputechconf.com