

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());

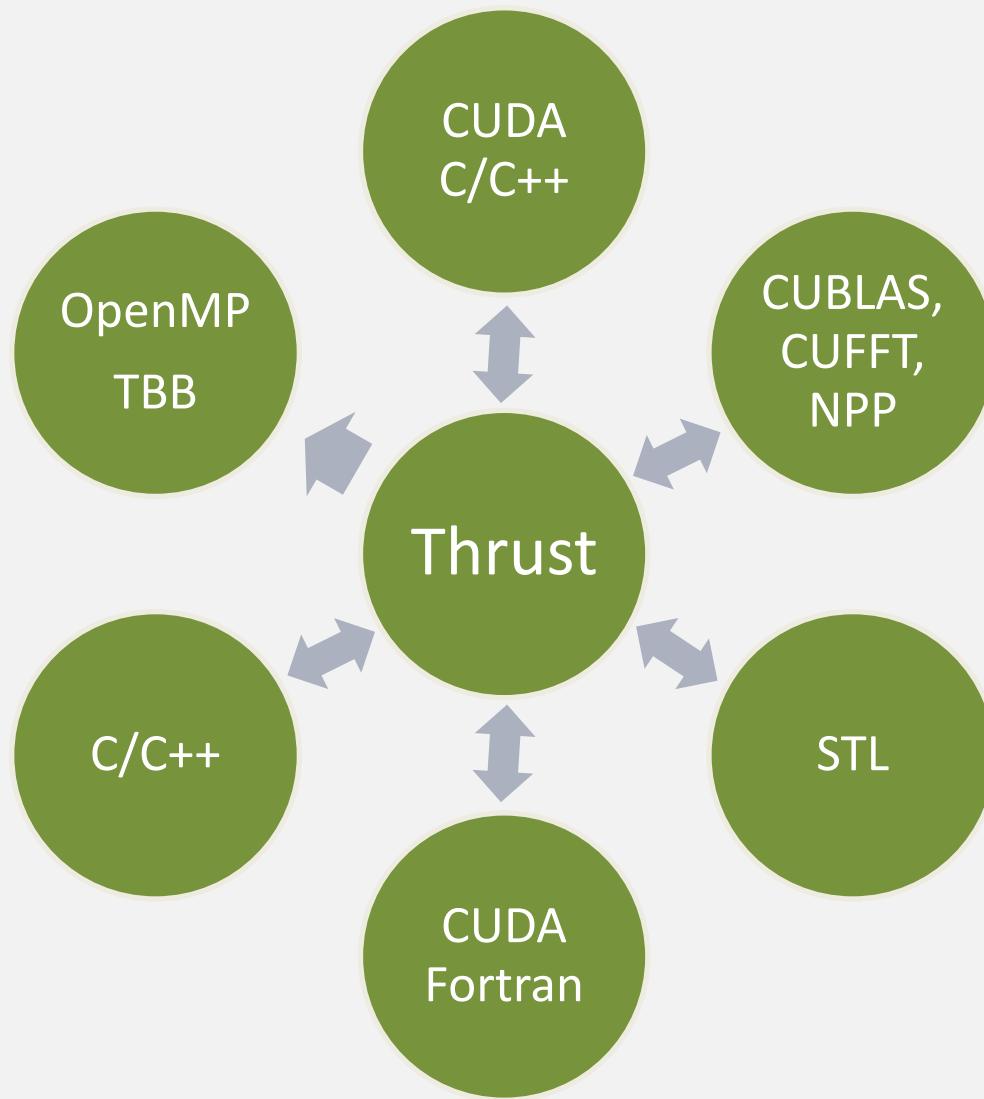
// memory automatically released
```

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

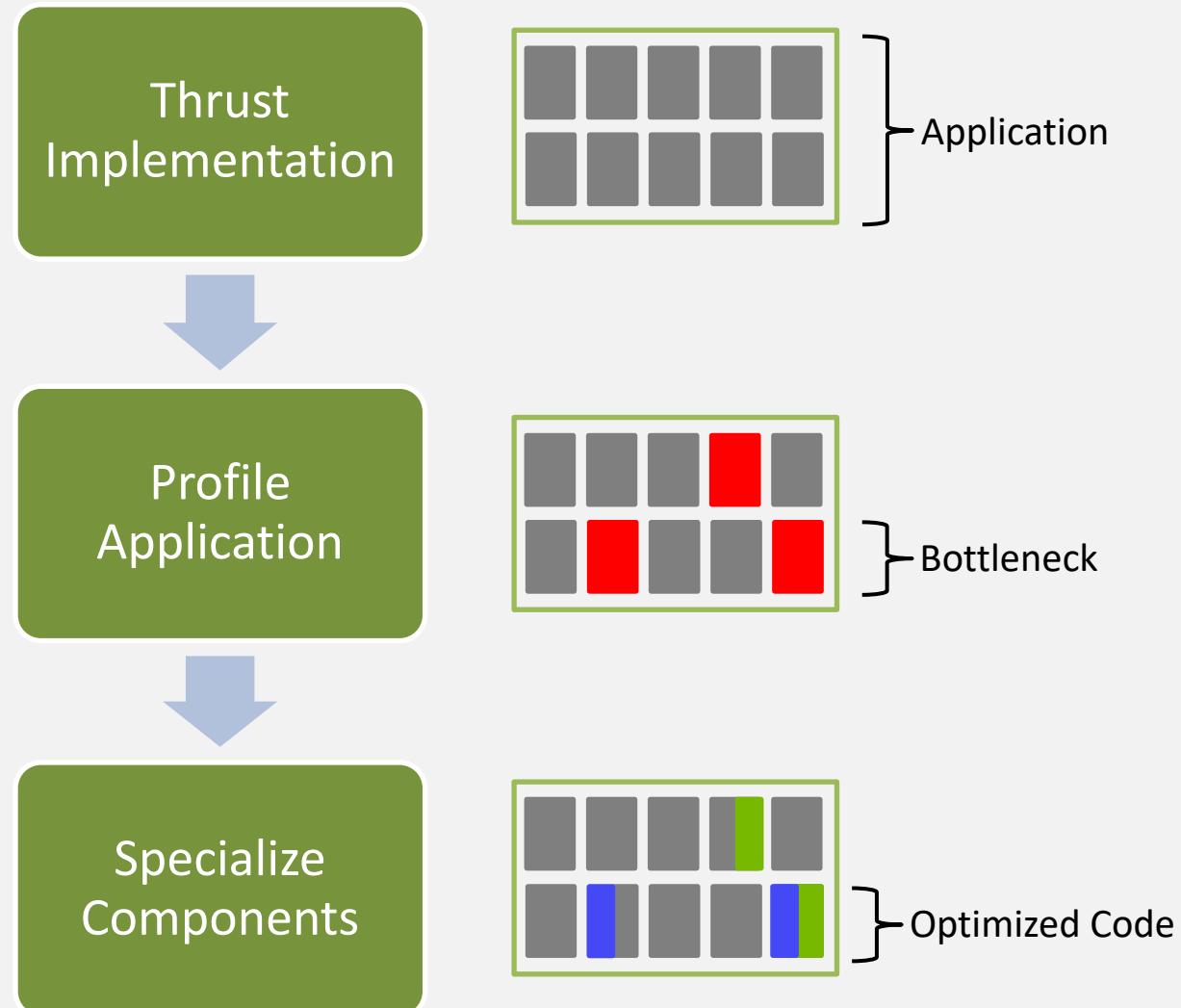
```
thrust::omp::vector<float> my_omp_vec(100) ;
thrust::cuda::vector<float> my_cuda_vec(100) ;

...
// reduce in parallel on the CPU
thrust::reduce(my_omp_vec.begin() , my_omp_vec.end()) ;

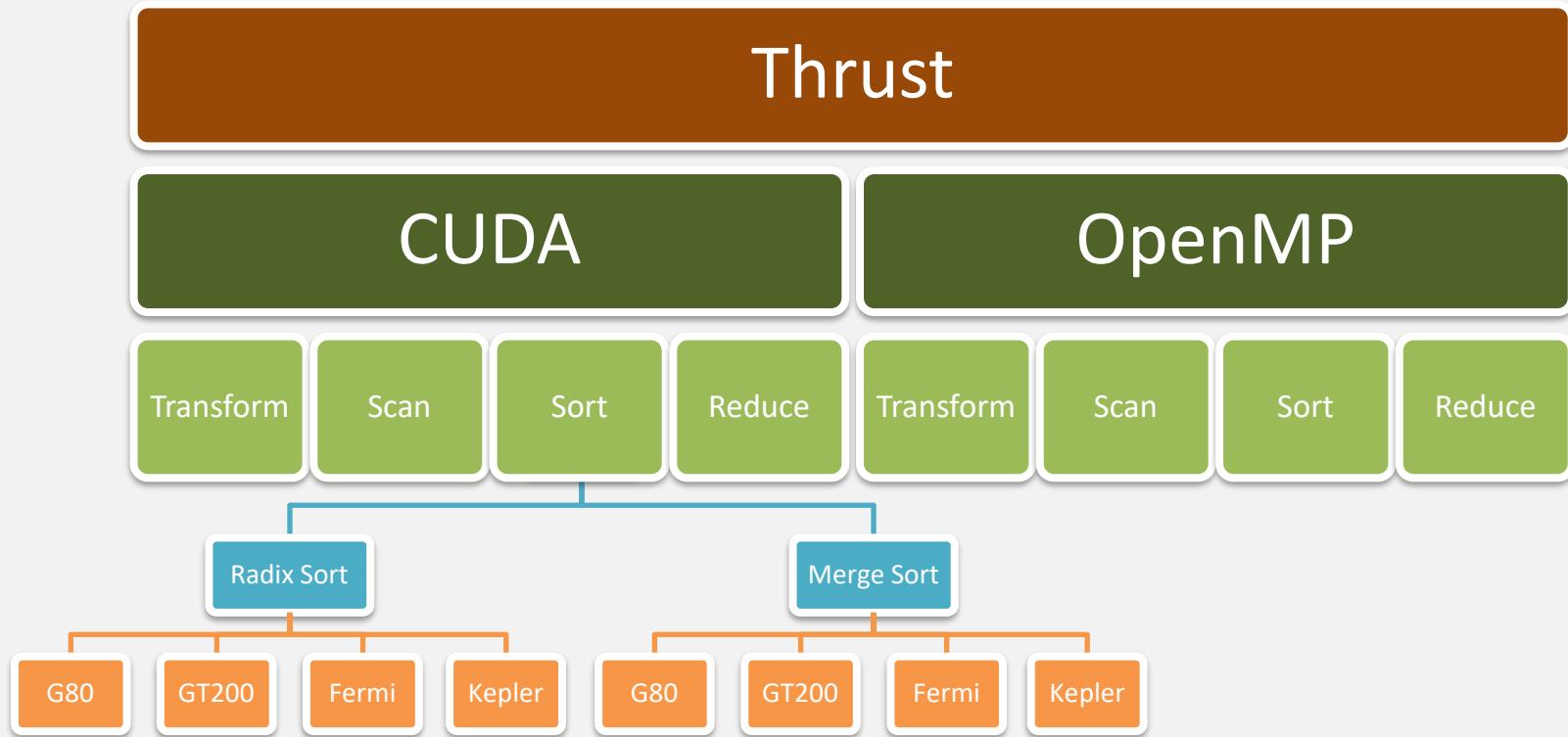
// sort in parallel on the GPU
thrust::sort(my_cuda_vec.begin() , my_cuda_vec.end()) ;
```

Potential Workflow

- Implement Application with Thrust
- Profile Application
- Specialize Components as Necessary



Performance Portability



Performance Portability

Slashdot NEWS FOR NERDS. STUFF THAT MATTERS.

▶ Stories  Recent Popular Search

Slashdot is powered by [your submissions](#), so send in your scoop

+ - Developers: Sorting Algorithm Breaks Giga-Sort Barrier, With GPUs

Posted by [timothy](#) on Sunday August 29, @10:22PM
from the quick-like-double-time dept.

An anonymous reader writes

"Researchers at the University of Virginia have recently open sourced an algorithm capable of sorting at a rate of [one billion \(integer\) keys per second using a GPU](#). Although GPUs are often assumed to be poorly suited for algorithms like sorting, their results are several times faster than the best known CPU-based sorting implementations."



[Read More...](#) |   | [99 comments](#) ▶ gpu graphics hardware developers programming story

+ - Your Rights Online: Network Neutrality Is Law In Chile

Posted by [timothy](#) on Sunday August 29, @07:25PM
from the muy-bien-tal-vez dept.

An anonymous reader writes

"Chile is the first country of the world [to guarantee by law the principle of network neutrality](#), according to the Telecommunications Market Commission's Blog from Spain. The official newspaper of the Chilean Republic published yesterday



Extensibility

- Customize temporary allocation
- Create new backend systems
- Modify algorithm behavior
- New in Thrust v1.6

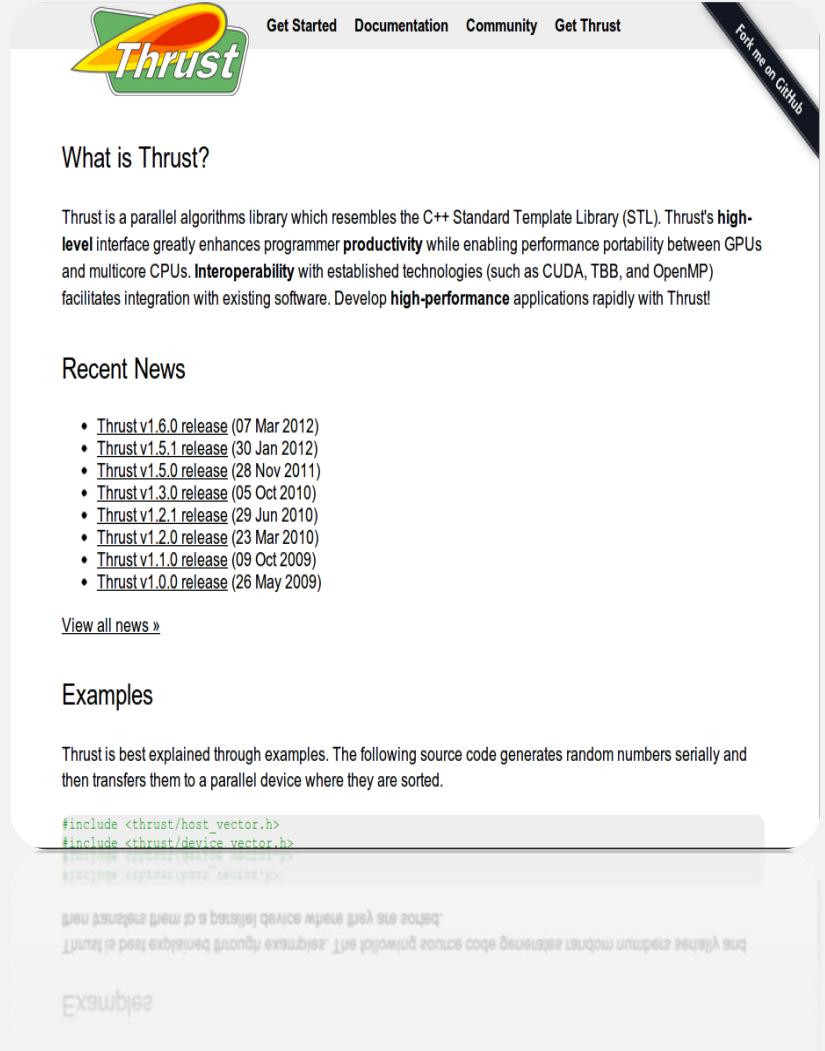
Robustness

- 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 homepage of the Thrust GitHub repository. At the top right is a "Fork me on GitHub" button. The header includes links for "Get Started", "Documentation", "Community", and "Get Thrust". The main content area features a large Thrust logo with a yellow and red swoosh. Below it is a section titled "What is Thrust?" which describes Thrust as a parallel algorithms library resembling the C++ Standard Template Library (STL). It highlights its high-level interface, productivity, performance portability between GPUs and multicore CPUs, interoperability with CUDA, TBB, and OpenMP, and integration with existing software. A "Recent News" section lists nine releases from 2009 to 2012, with a link to "View all news ». Below that is an "Examples" section with a snippet of C++ code demonstrating random number generation and sorting. A note at the bottom indicates that the code is released under the Apache License.

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.

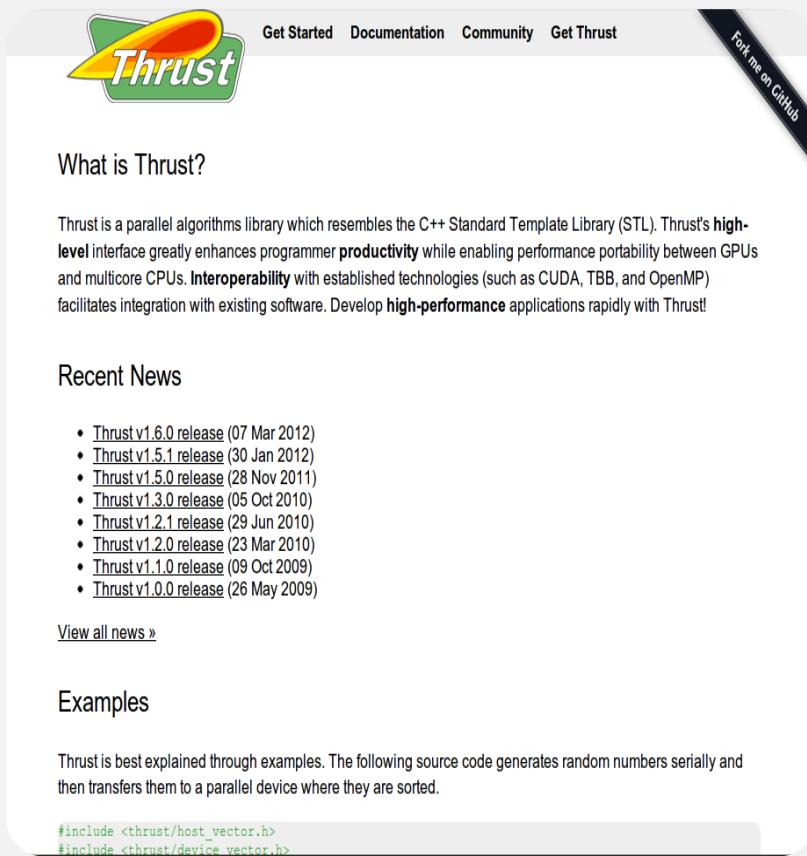
```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <random>
#include <vector>
```

Apache License Version 2.0, January 2004
http://www.apache.org/licenses/

Resources

- Documentation
- Examples
- Mailing List
- Webinars
- Publications

thrust.github.com



The screenshot shows the homepage of the Thrust GitHub repository. At the top, there's a navigation bar with links for "Get Started", "Documentation", "Community", and "Get Thrust". To the right of the navigation is a "Fork me on GitHub" button. The main content area features a large green logo with the word "Thrust" in white. Below the logo, the heading "What is Thrust?" is followed by a paragraph explaining that Thrust is a parallel algorithms library resembling the C++ Standard Template Library (STL). It highlights its productivity-enhancing interface, performance portability between GPUs and multicore CPUs, interoperability with CUDA, TBB, and OpenMP, and its ability to facilitate integration with existing software for high-performance applications. A section titled "Recent News" lists several releases from 2009 to 2012, each with a link and date. A "View all news »" link is also present. Another section titled "Examples" shows a snippet of C++ code for generating random numbers serially and transferring them to a parallel device for sorting. A note at the bottom of this section states: "Please use your own source code instead of my示例代码. Some example files may be lost." The footer contains the text "See more examples" and "See more examples" again.

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>
```

Please use your own source code instead of my示例代码. Some example files may be lost.