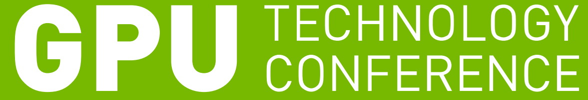
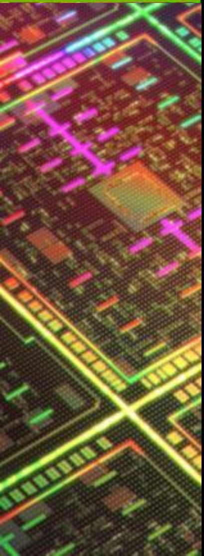

GPU
TECHNOLOGY
CONFERENCE

Data Parallel Programming with Patterns
Peng Wang, Developer Technology, NVIDIA

Overview

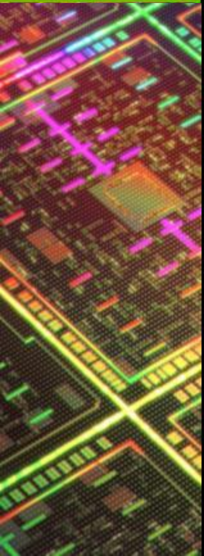
- Patterns in Data Parallel Programming

- Examples
  - Radix sort
  - Cell list calculation in molecular dynamics
  - MPI communication calculation in particle code



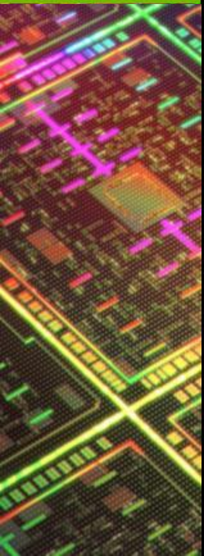
GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE



Data Parallel Programming Model

- Many independent threads of execution
  - All running the same program
- Threads operate in parallel on separate inputs
  - Produce an output per input
- Works well when outputs depend on small, bounded input

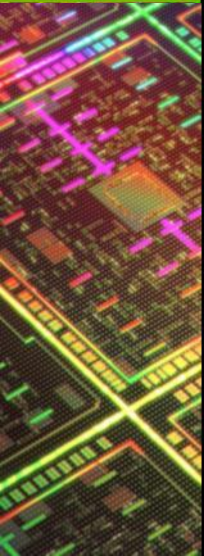


Parallel Programming with Patterns

- Traditionally, design and programming can be made easier by pattern

- Make parallel programming easier by recognizing the underlying pattern

- Using parallel primitive operations to express your algorithm
  - Reuse existing high performance implementations
  - Ease to maintain, scale, etc.

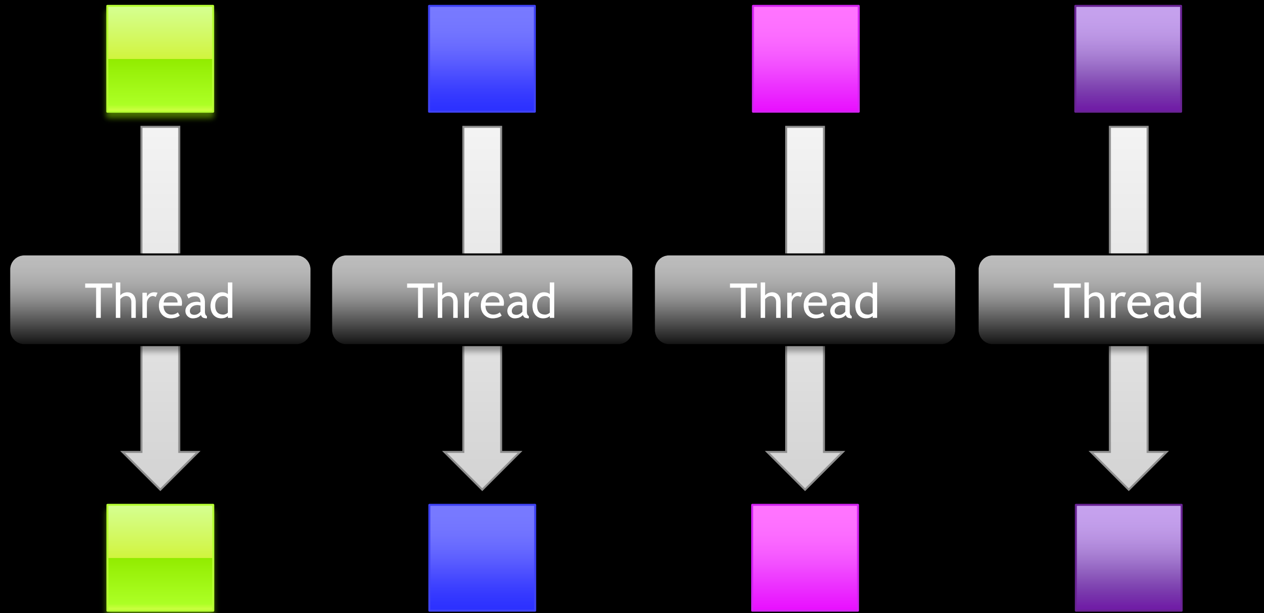


Common Patterns

- Map
- Stencil
- Reduce
- Scan
- Gather/scatter
- Compaction
- Partition

GPU TECHNOLOGY CONFERENCE

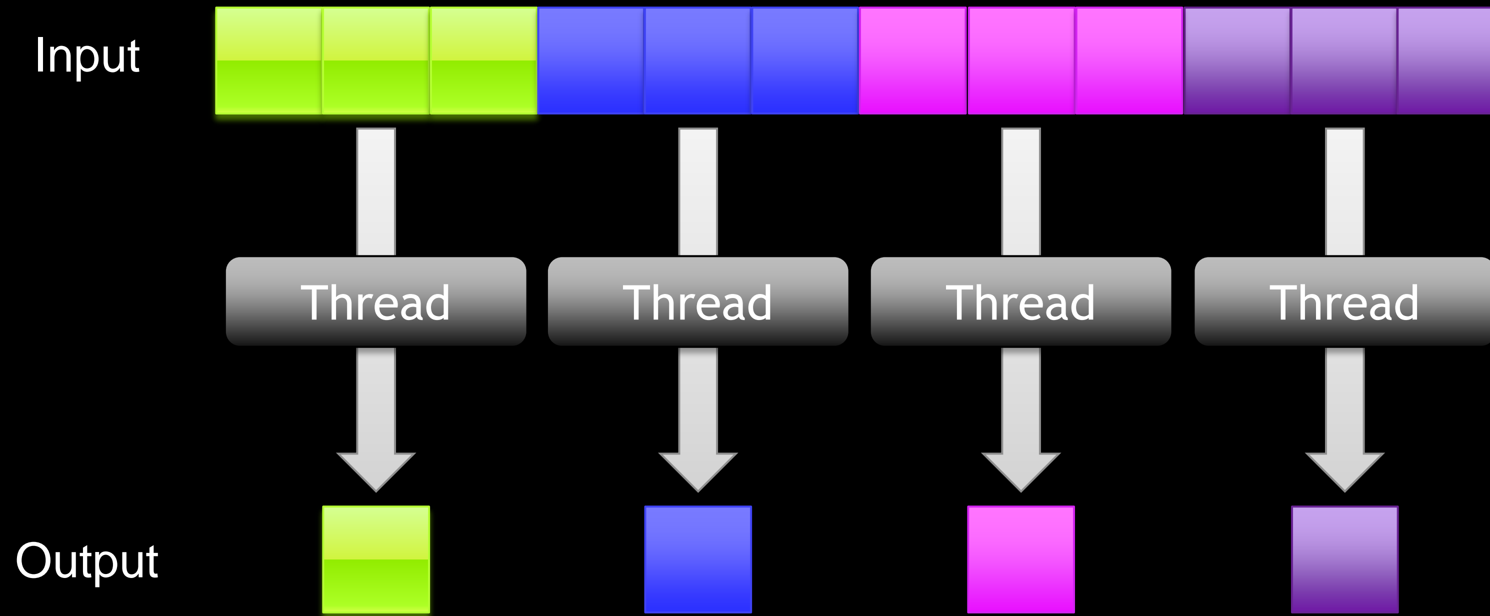
Map



- One-to-one Input-output dependence (e.g., scalar)

GPU TECHNOLOGY CONFERENCE

Stencil

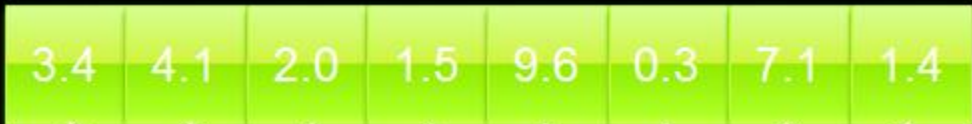


- Local neighborhood input-output dependence

GPU TECHNOLOGY CONFERENCE

Reduce

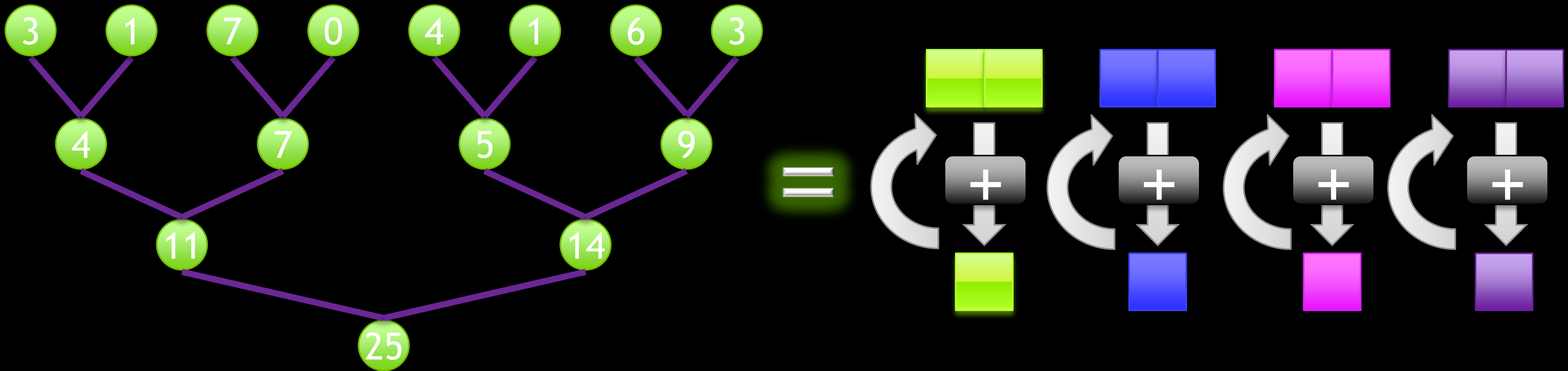
Input



Output

GPU TECHNOLOGY CONFERENCE

Parallel Reduction: Easy

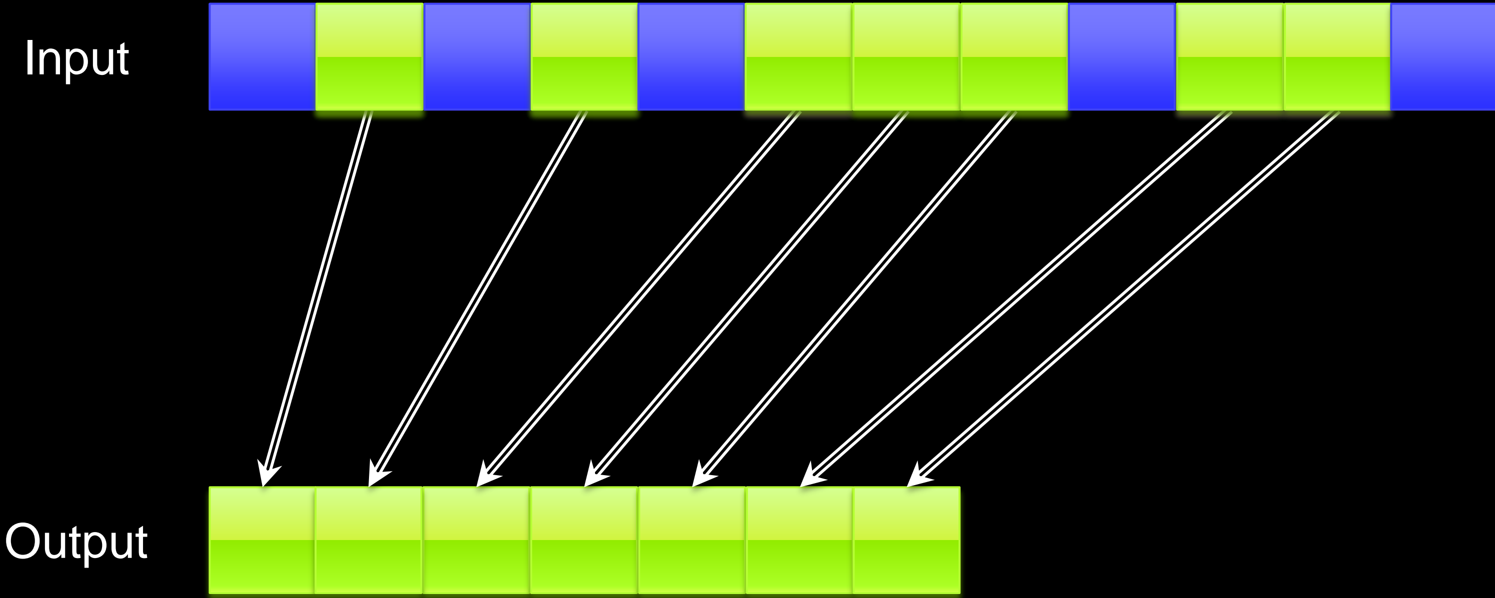


- Repeated local neighborhood access: O(log n) reps
  - Static data dependences, uniform output

GPU TECHNOLOGY CONFERENCE

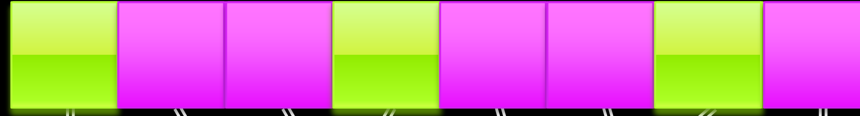
Compact

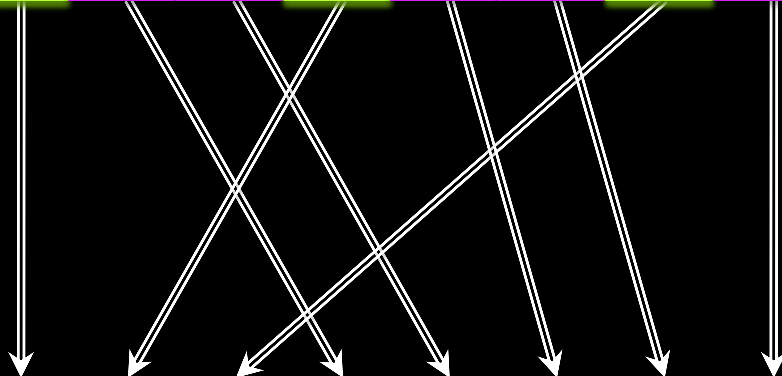
- Remove unneeded or invalid elements (blue)



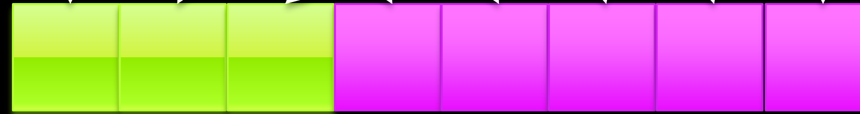
Partition

Input





Output

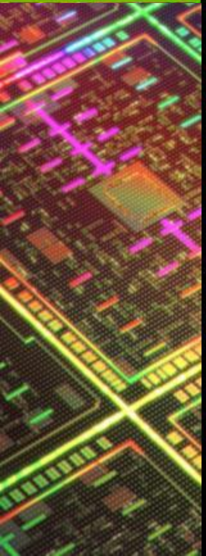


- Example: radix sort, building trees, cell-list, etc

GPU TECHNOLOGY CONFERENCE

“Where do I write my output?”

- Partition, compact and allocate require all threads to answer
- The answer is:
  “That depends on how much the other threads output!”
- “Scan” is an efficient, parallel way to answer this question



GPU TECHNOLOGY CONFERENCE

Scan (Prefix Sums)

- Given array $A = [a_0, a_1, \ldots, a_{n-1}]$ and a binary associative operator $\oplus$ with identity I,

  scan(A) = $[I, a_0, (a_0 \oplus a_1), \ldots, (a_0 \oplus a_1 \oplus \ldots \oplus a_{n-2})]$ (exclusive)

  scan(A) = $[a_0, (a_0 \oplus a_1), \ldots, (a_0 \oplus a_1 \oplus \ldots \oplus a_{n-1})]$ (inclusive)

- Example: if $\oplus$ is +, then

  Scan([3 1 7 0 4 1 6 3]) = [0 3 4 11 11 15 16 22] (exclusive)

  Scan([3 1 7 0 4 1 6 3]) = [3 4 11 11 15 16 22 25] (inclusive)

S. Sengupta, M. Harris, Y. Zhang, and J.D. Owens.
"Scan Primitives for GPU Computing". *Graphics Hardware 2007*

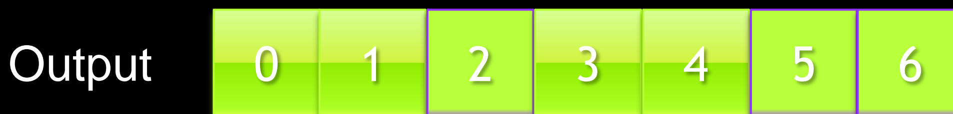
GPU TECHNOLOGY CONFERENCE

Applications of Scan

- A simple and useful building block for many parallel apps:
  - Compaction
  - Radix sort
  - Quicksort (segmented scan)
  - String comparison
  - Lexical analysis
  - Stream compaction
  - Run-length encoding
  - Allocation
  - Polynomial evaluation
  - Solving recurrences
  - Tree operations
  - Histograms
  - Summed area tables
  - And many more!
- (Interestingly, scan is unnecessary in sequential computing)

Compact using Scan

- Flag unneeded elements with zero:



- Threads with flag == 1 use scan result as address for output:
- Implementation using Scan+Scatter

GPU TECHNOLOGY CONFERENCE

Radix Sort / Partition using Scan

GPU TECHNOLOGY CONFERENCE

Current Digit: 0

	0	1	2	3	4	5	6	7
i = index	0	1	2	3	4	5	6	7
b = current bit	010	001	111	011	101	011	000	110
d = invert b	1	0	0	0	0	0	1	1
f = scan(d)	0	1	1	1	1	1	1	2
t = numZeros + i – f	3	3	4	5	6	7	8	8
out = b ? t : f	0	3	4	5	6	7	1	2
	010	000	110	001	111	011	101	011

numZeros=3

Radix Sort / Partition using Scan

	0	1	2	3	4	5	6	7
b = current bit	010	000	110	001	111	011	101	011
d = invert b	0	1	0	1	0	0	1	0
f = scan(d)	0	0	1	1	2	2	2	3
t = numZeros + i – f	3	4	4	5	5	6	7	7
d = b ? t : f	3	0	4	1	5	6	2	7
	000	001	101	010	110	111	011	011

Header row: i = index

Current Digit: 1

numZeros=3

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

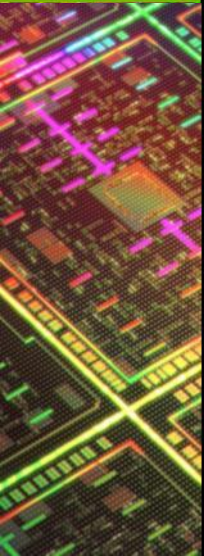
Radix Sort / Partition using Scan

i = index	0	1	2	3	4	5	6	7
b = current bit	000	001	101	010	110	111	011	011
d = invert b	1	1	0	1	0	0	1	1
f = scan(d)	0	1	2	2	3	3	3	4
t = numZeros + i – f	5	5	5	6	6	7	8	8
d = b ? t : f	0	1	5	2	6	7	3	4
	0	1	2	3	3	5	6	7

Current Digit: 2

numZeros=5

GPU TECHNOLOGY CONFERENCE



No need to re-implement

- Open source libraries under active development

- CUDPP: CUDA Data-Parallel Primitives library
  - http://code.google.com/p/cudpp (BSD License)
- Thrust
  - http://code.google.com/p/thrust (Apache License)

GPU TECHNOLOGY CONFERENCE

CUDPP

- C library of high-performance parallel primitives for CUDA
  - M. Harris (NVIDIA), J. Owens (UCD), S. Sengupta (UCD), A. Davidson (UCD), S. Tzeng (UCD), Y. Zhang (UCD)
- Algorithms
  - cudppScan, cudppSegmentedScan, cudppReduce
  - cudppSort, cudppRand, cudppSparseMatrixVectorMultiply
- Additional algorithms in progress
  - Graphs, more sorting, trees, hashing, autotuning

GPU TECHNOLOGY CONFERENCE

CUDPP Example

```
CUDPPConfiguration config = { CUDPP_SCAN,
  CUDPP_ADD, CUDPP_FLOAT, CUDPP_OPTION_FORWARD };

CUDPPHandle plan;
CUDPPResult result = cudppPlan(&plan, config,
 numElements, 1, 0);
cudppScan(plan, d_odata, d_idata, numElements);
```

GPU TECHNOLOGY CONFERENCE

Thrust

- C++ template library for CUDA
  - Mimics Standard Template Library (STL)
- Containers
  - thrust::host_vector<T>
  - thrust::device_vector<T>
- Algorithms
  - thrust::sort()
  - thrust::reduce()
  - thrust::inclusive_scan()
  - Etc.

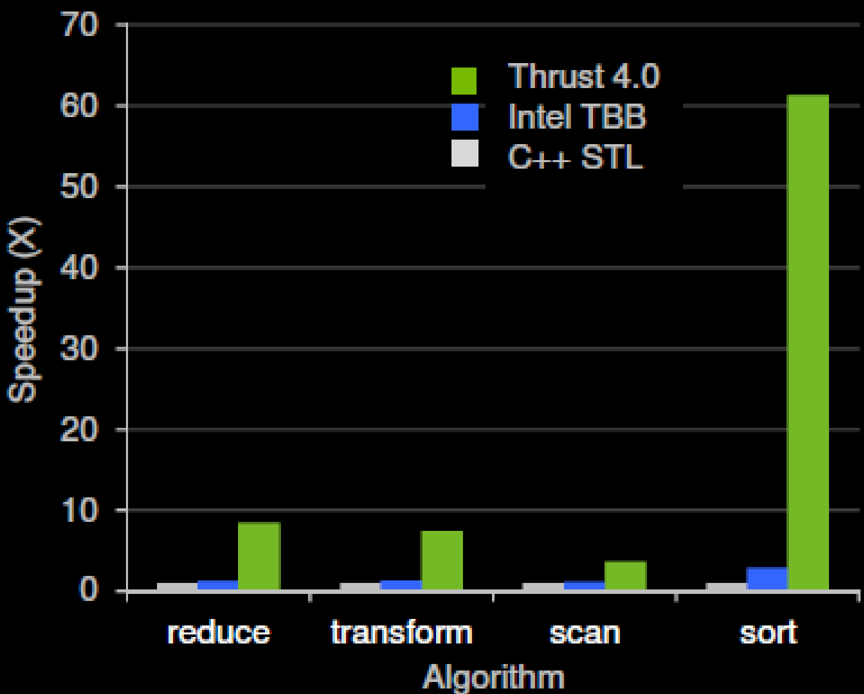
Thrust Example

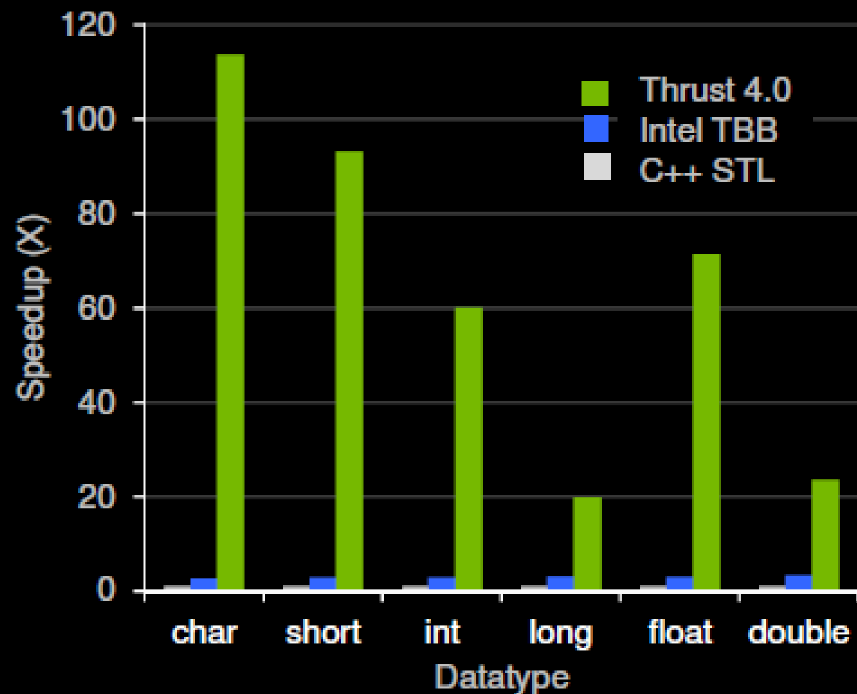
```
// generate 16M random numbers on the host
thrust::host_vector<int> h_vec(1 << 24);
thrust::generate(h_vec.begin(), h_vec.end(), rand);

// 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());
```

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

Thrust Performance





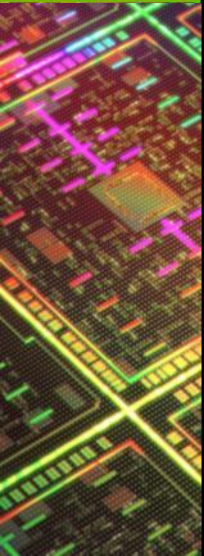
* Thrust 4.0, NVIDIA Tesla C2050 (Fermi)

* Core i7 950 @ 3.07GHz

Example 1: Build Cell-List in Particle Code

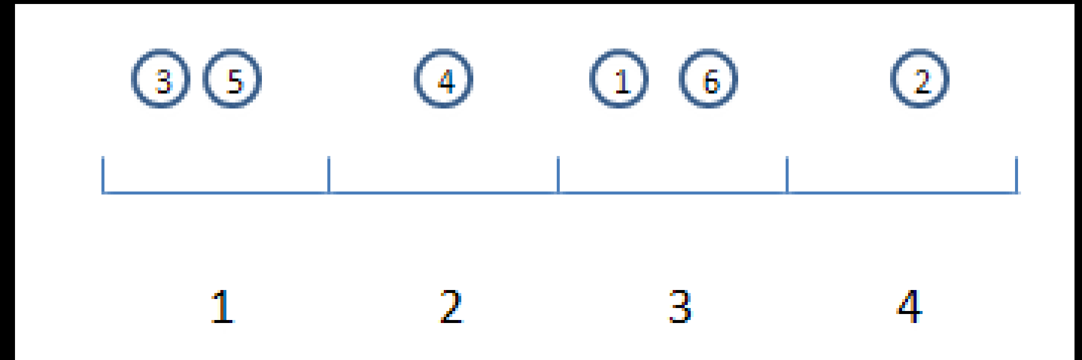
- Short range force calculation
  - Naïve O(N^2) N-body like algorithm: not practical for large problems
- Avoiding searching all other particles
  - Cell-list (link-list)
  - Neighbor-list
- LAMMPS

GPU TECHNOLOGY CONFERENCE

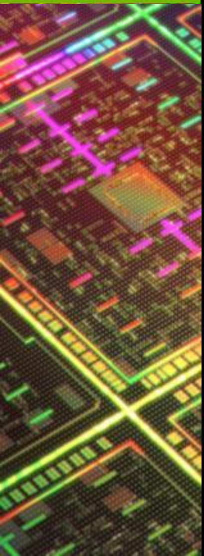


Cell-list

- Decompose the rectangular domain into cells
  - When calculating force, every particle only needs to look at the 27 neighboring cells, saving the cost of looking at all other particles
- Each cell has list of particles belonging to it
  - Partition Pattern



GPU TECHNOLOGY CONFERENCE



Data-structure for cell-list

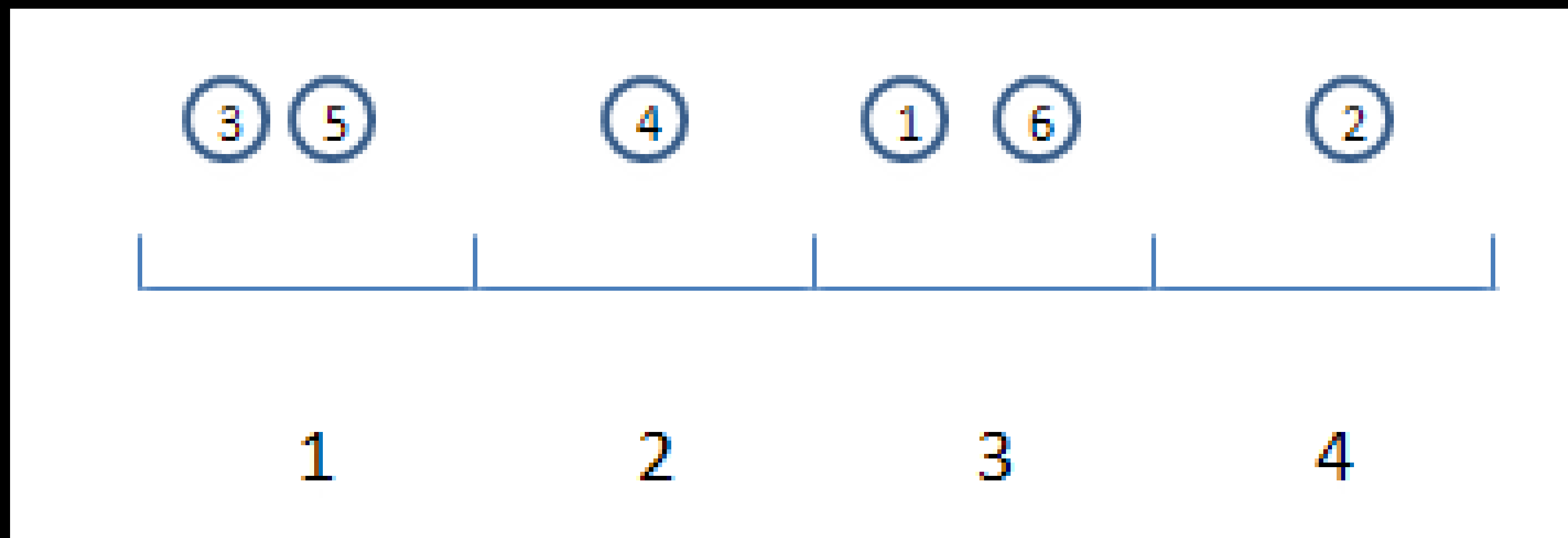
- Two arrays

```
typedef struct {
   int *cell_list;
   int *cell_counts;
} cell_list_gpu;
```

  - cell_list[nall]: list particle id belonging to each cell in a packed way
  - cell_counts[ncell+1]: list starting position of each cell in the cell_list array

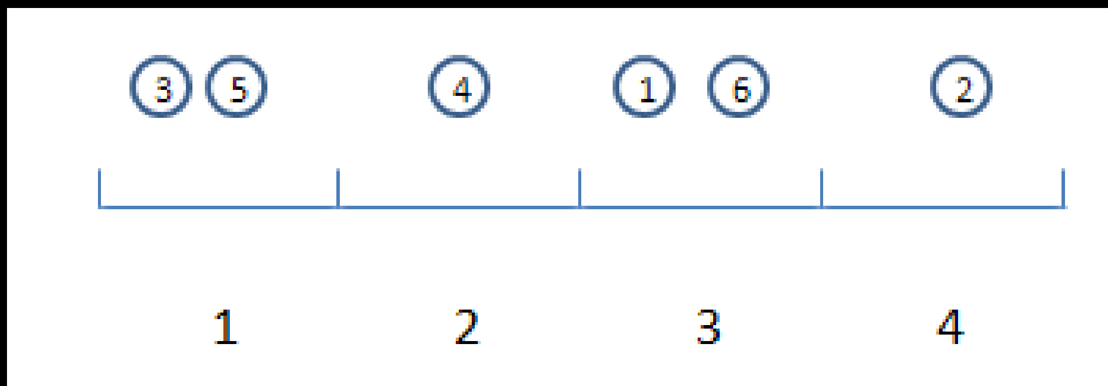
Cell-list build algorithm

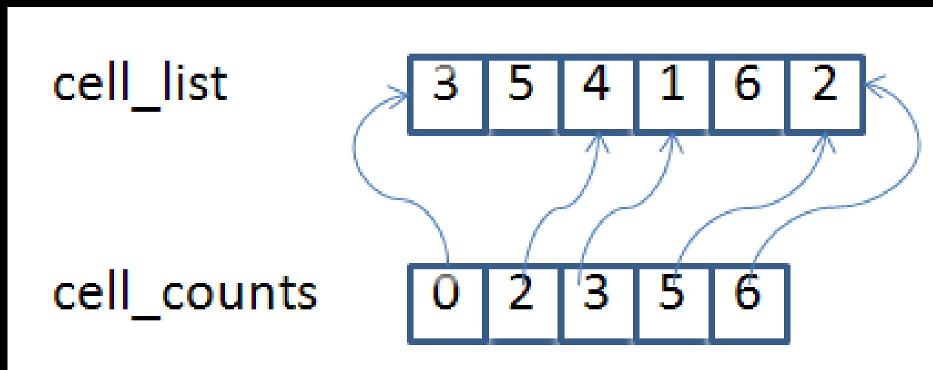
- E.g.



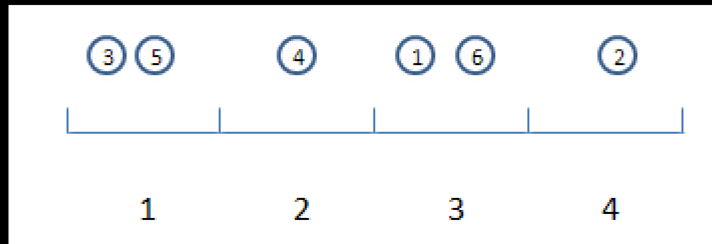
GPU TECHNOLOGY CONFERENCE

Cell-list build algorithm

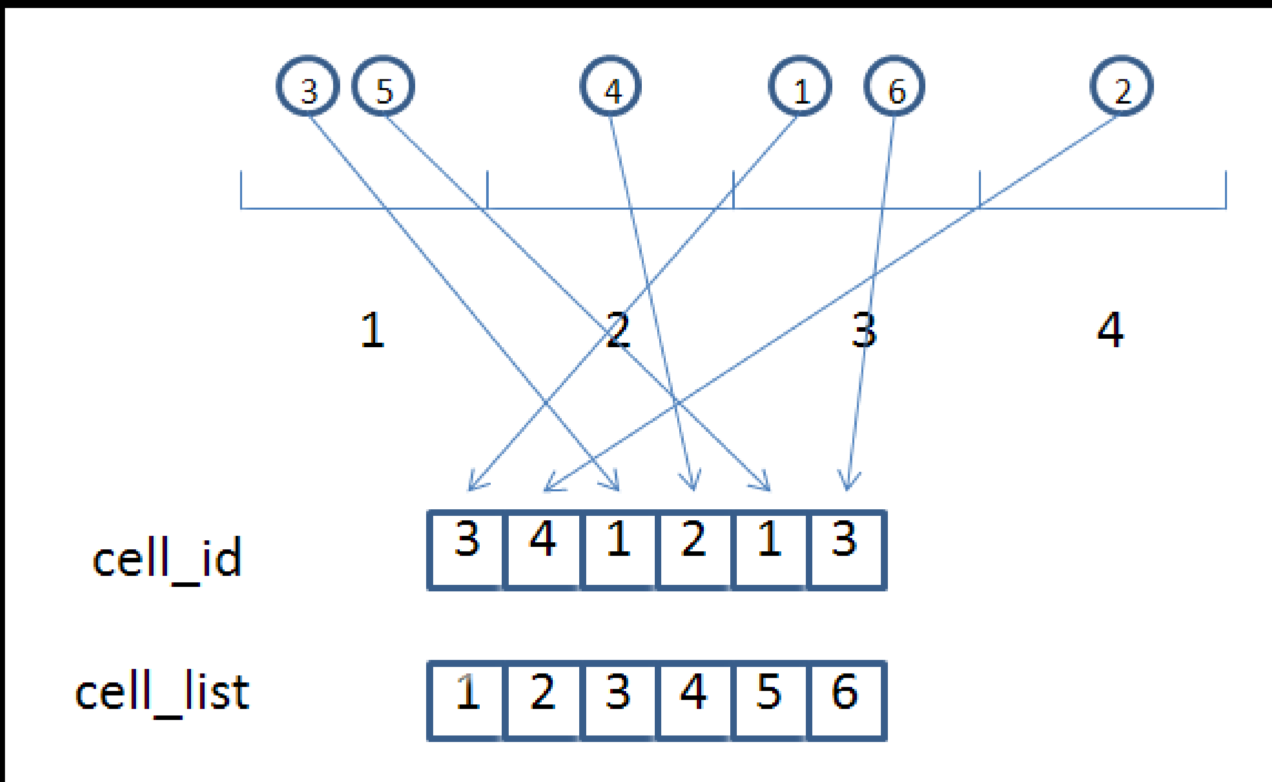




Cell-list build algorithm



- Calculate the cell id of each particle, store the cell id to a temporary array cell_id, particle id to cell_list

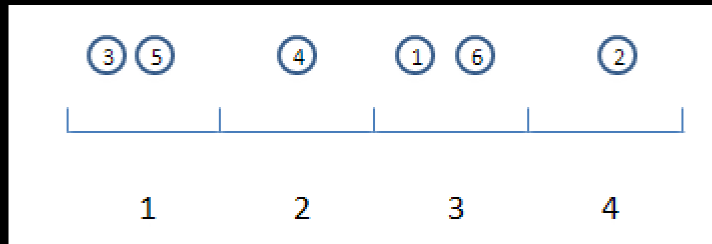


One thread per particle
Embarassingly parallel
Fully coalesced R/W

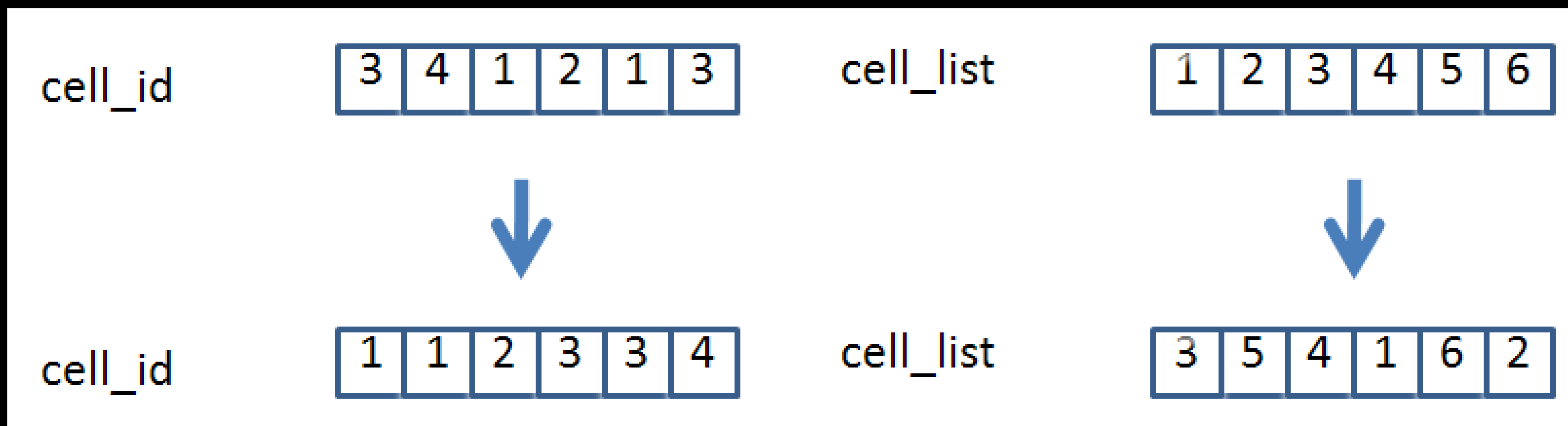
GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

Cell-list build algorithm

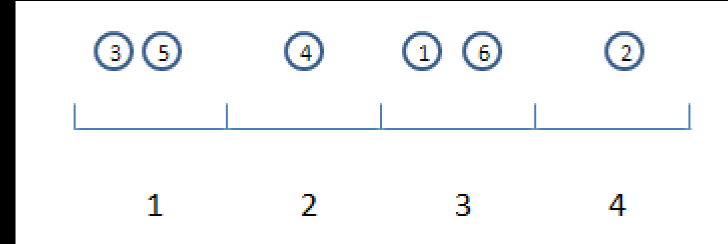


- Sort using cell_id as the key and cell_list as the value
  - Use CUDPP radix sort
  - cell_list then has the correct order

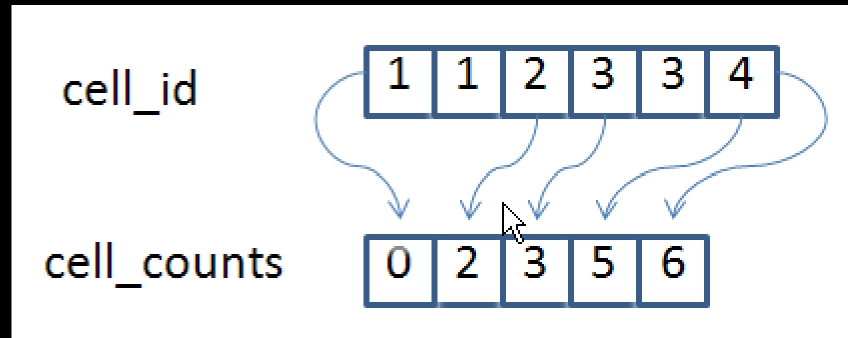


GPU TECHNOLOGY CONFERENCE

Cell-list build algorithm



- Calculate cell_counts from the sorted cell_id
  - Compaction
  - One thread per particle
  - Compare cell id in the left and itself, if different, that's a cell boundary
  - Adding two special cases to handle the two boundaries



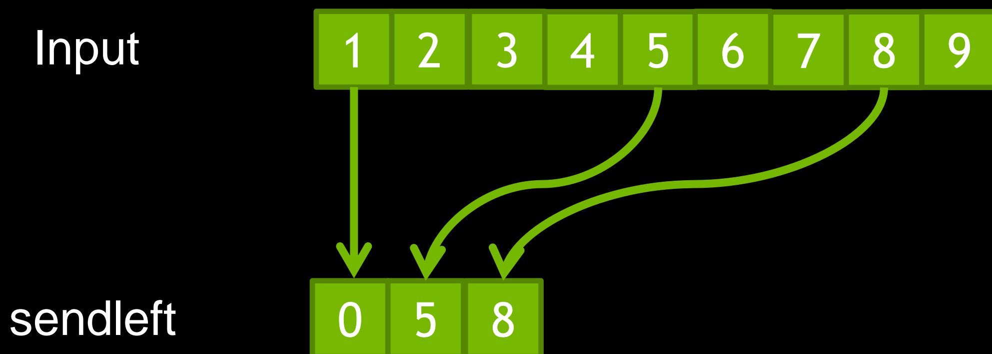
GPU TECHNOLOGY CONFERENCE

Example 2: MPI Communication Calculation

- MPI communication in particle codes
  - Irregular particle position
  - Need to calculate outgoing particles at each time step
- Naïve implementation
  - Send all particle location to CPU to do the calculation
  - Potentially a large overhead in data transfer
- Data parallel implementation: compaction pattern

Problem

- Input:
  - pos[N]: particle position
- Ouput:
  - sendleft[mleft]: mleft particles that needs to be send to left
  - sendright[mright]: mright particles that needs to be send to right



Note: Copying non-continuous boundary to a buffer in structured grid is a special case of this

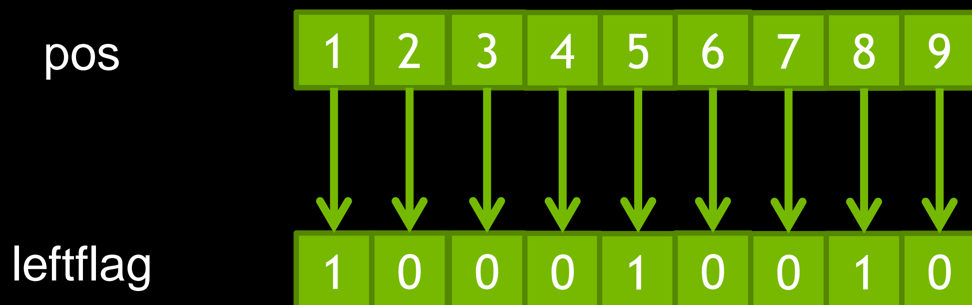
GPU TECHNOLOGY CONFERENCE

Code

```
struct find_left
{
  __host__ __device__ int operator()(const float &a) const
  {
    if (a < LEFT_BOUNDARY)
      return 1;
    else
      return 0;
  }
};


thrust::transform(pos, pos+me, leftflag, find_left());
thrust::exclusive_scan(leftflag, leftflag+me , leftpos);
msendleft  = leftpos(me-1) + leftflag(me-1);
thrust::device_vector<float> sendleft(msendleft);
thrust::scatter_if(pos, pos+me, leftpos, leftflag, sendleft, is_true());
```

Code Walk-through

- Flag particles that need to be sent left




```
thrust::transform(pos, pos+me, leftflag, find_left());
```

GPU TECHNOLOGY CONFERENCE

Code Walk-through

- Scan to find out their output locations

leftflag

1	0	0	0	1	0	0	1	0

leftpos

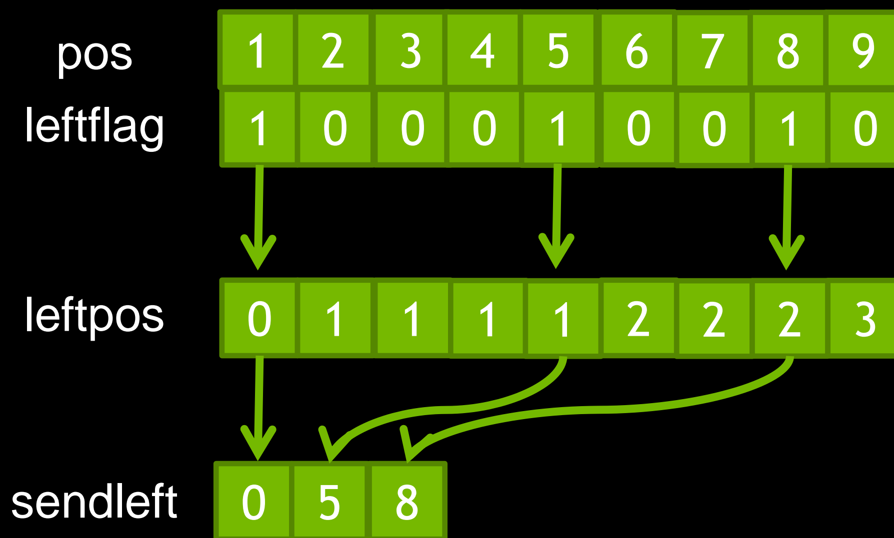
0	1	1	1	1	2	2	2	3

```
thrust::exclusive_scan(leftflag, leftflag+me , leftpos);
msendleft  = leftpos(me-1) + leftflag(me-1);
thrust::device_vector<float> sendleft(msendleft);
```

GPU TECHNOLOGY CONFERENCE

Code Walk-through

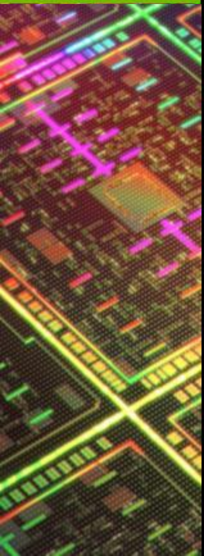
- Scatter to sendleft array




```
thrust::scatter_if(pos, pos+me, leftpos, leftflag, sendleft, is_true());
```

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE



Conclusion

- Many parallel problems can be solved using parallel primitives
- Think parallel
  - Use data parallel primitives to express your algorithm
  - Partition pattern is especially common: recognize it and find the best way to implement it
- NVIDIA provides you high-level tools on parallel primitives
  - Don’t re-invent the wheels