GPU TECHNOLOGY CONFERENCE

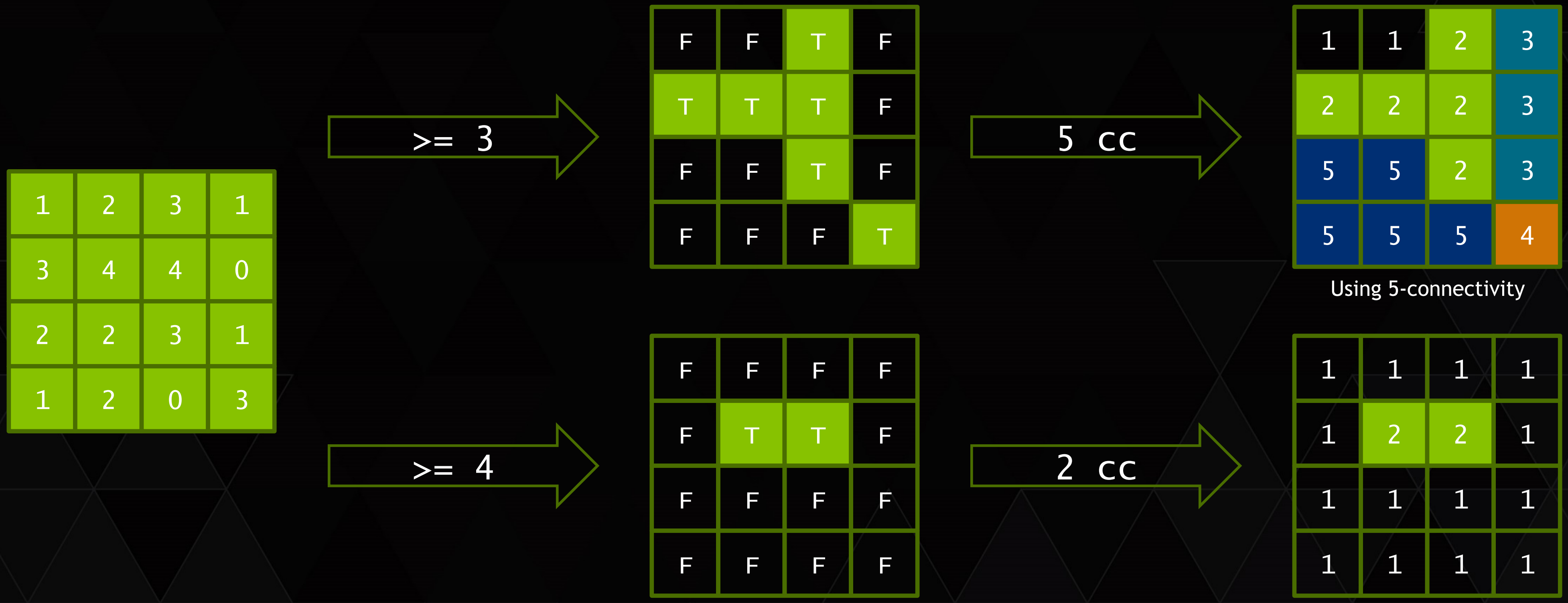
BINARY SEGMENTATION OF MANY 3D CUBES IN CUDA

JULIEN DEMOUTH, NVIDIA

GPU TECHNOLOGY CONFERENCE

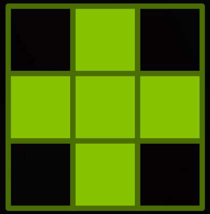
PROBLEM IN 2D

- A grid of values + list of threshold(s), label the connected components

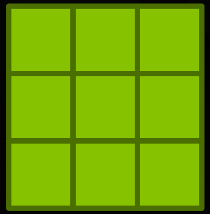


PROBLEM

- 2 types of connectivity



5 points



9 points

- Generalized to 3D with 7-point and 27-point connectivity

GPU TECHNOLOGY CONFERENCE

CONNECTED COMPONENTS IN GRAPHS

- Graph labeling using BFS on the GPU
  - See Duane Merrill and Michael Garland presentation
  - http://on-demand.gputechconf.com/gtc-express/2013/presentations/understanding-parallel-graph-algorithms.pdf
- The Gunrock library for connected components in graphs
  - https://github.com/gunrock/gunrock
- But we want to solve a much more structured problem with a 3D cube

OUR STRATEGY

- Each cell is given a different label at the beginning
- We propagate the “min”-labels to the right, then to the left

```
l = label[0], t = tag[0]
for i = 1 to n-1:
  if t == tag[i] and l <= label[i]:
    label[i] = l
  elif t == tag[i]:
    l = label[i]
  else:
    l = label[i], t = tag[i]
for i = n-2 to 0:
  ...
```

OUR STRATEGY

- Extend it to 2D (assume 9-point connectivity)

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23

Sweep 1st row

0	0	0	3	3	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23

Propagate 1st row

0	0	0	3	3	5	6	7
8	9	0	0	3	5	6	7
16	17	18	19	20	21	22	23

Sweep 2nd row

0	0	0	3	3	5	6	7
8	8	0	0	3	5	6	7
16	17	18	19	20	21	22	23

Propagate 2nd row

0	0	0	3	3	5	6	7
8	8	0	0	3	5	6	7
16	0	0	3	0	5	6	7

Sweep 3rd row

0	0	0	3	3	5	6	7
8	8	0	0	3	5	6	7
0	0	0	3	0	0	6	7

...

OUR STRATEGY

- If we stop now, we have incorrect cells

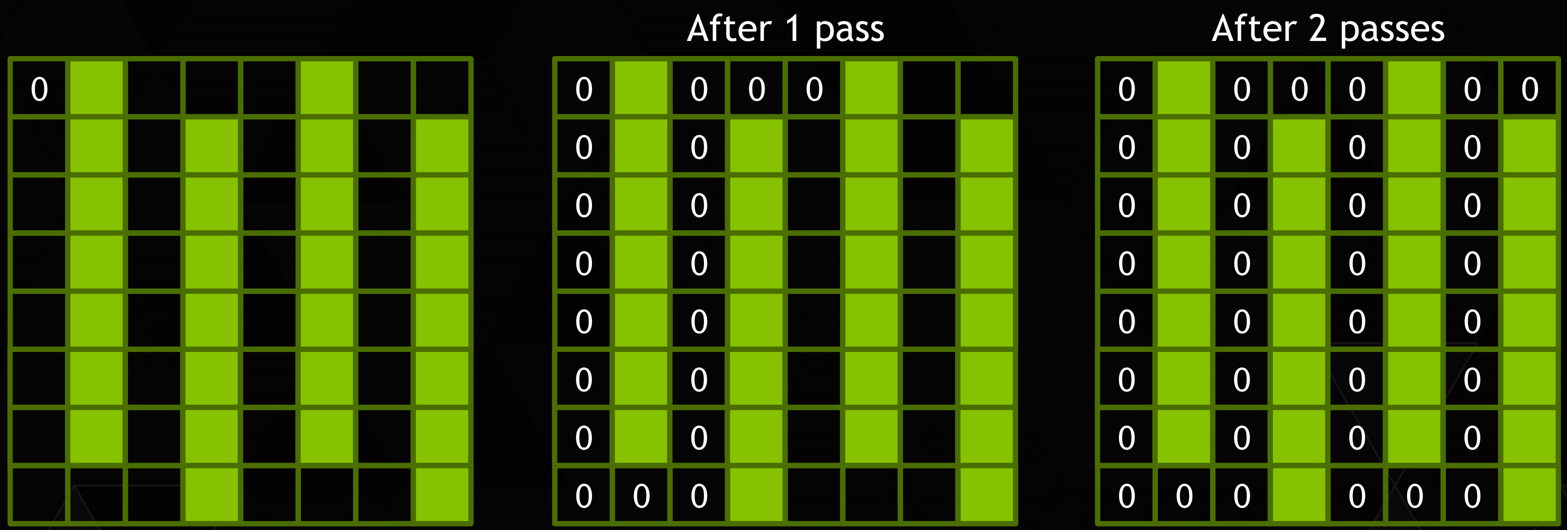
0	0	0	3	3	5	6	7
8	8	0	0	3	5	6	7
0	0	0	3	0	0	6	7

- We have to sweep "up" to bring the "bottom 0" to the "top"

0	0	0	3	3	0	6	7
8	8	0	0	3	0	6	7
0	0	0	3	0	0	6	7

OUR STRATEGY

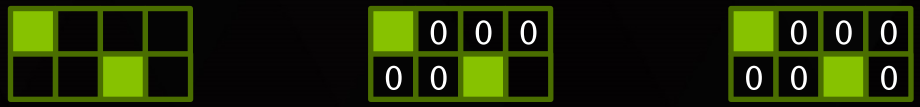
- How many passes do we need?



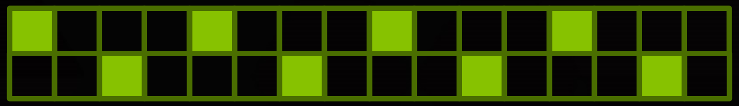
- Each connected component expands by at least one cell per pass: O(N^2)

OUR STRATEGY

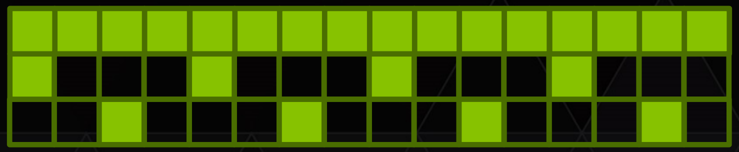
- We can build a lower bound in Ω(N^2)
- We use “up-and-down” blocks



- It takes two passes to propagate into an “up-and-down” block
- We can combine ~ N/4 such blocks per row



- We can build N/3 rows (add 1 row of insulator)



GPU TECHNOLOGY CONFERENCE

OUR STRATEGY

- In practice, it does not seem to happen much
  - To do: Evaluate the probabilistic complexity (à la quicksort)
- We use a simple stopping criterion
  - If no label changes during a pass, we have converged and we stop
- We trivially extent the strategy to 3D cubes
  - Use forward/backward passes in the Z dimension

IMPLEMENTATION

- How can we make the 1D pass fast?

- Use one warp (32 threads) per row
- Each thread stores 8 labels (for N = 256) in 8 registers (1 reg. per label)

- Do intra-warp segmented scan to get the values from the other threads

- It is implemented with __shfl and without shared memory
  - We use 12 __shfl per pass (6 for forward and 6 for backward)

IMPLEMENTATION

- The 2D propagation is done with one warp per 2D slice
- Each warp starts at row 0
- For each row
  - Threads in the warp get the values from the previous row
  - Do the 1D pass
- The 3D propagation alternates between the Y and Z dimensions

IMPLEMENTATION

- In a preprocessing step, we generate as many cubes as thresholds
- We pack the labels with the T/F tag
  - `int label = (z*N*N + y*N + x) | (value >= threshold) << 31`
- We reorganize the labels in memory to have perfectly coalesced accesses
- For a given row, store labels 0, 1, 2, 3, 8, …, 11, 16, …, 19
  - Thread 0 can read 0, 1, 2, 3 with a single int4
  - The warp can load all the labels in 2x perfectly coalesced int4 requests

GPU TECHNOLOGY CONFERENCE

PERFORMANCE RESULTS

- 3D cubes, 27-point connectivity
- Segmented CT reconstructions, 64 thresholds

Cube side	Tesla K20c (Time)	Tesla K80 (Time)	Passes
128	0.084s	0.046s	11
256	0.742s	0.364s	11

- No ECC
- Tesla K20c @ 2600MHz/758MHz, Tesla K80 @ 2505MHz/875MHz

Input CT scans from the RabbitCt benchmark: http://www5.cs.fau.de/research/projects/rabbitct/

PERFORMANCE RESULTS

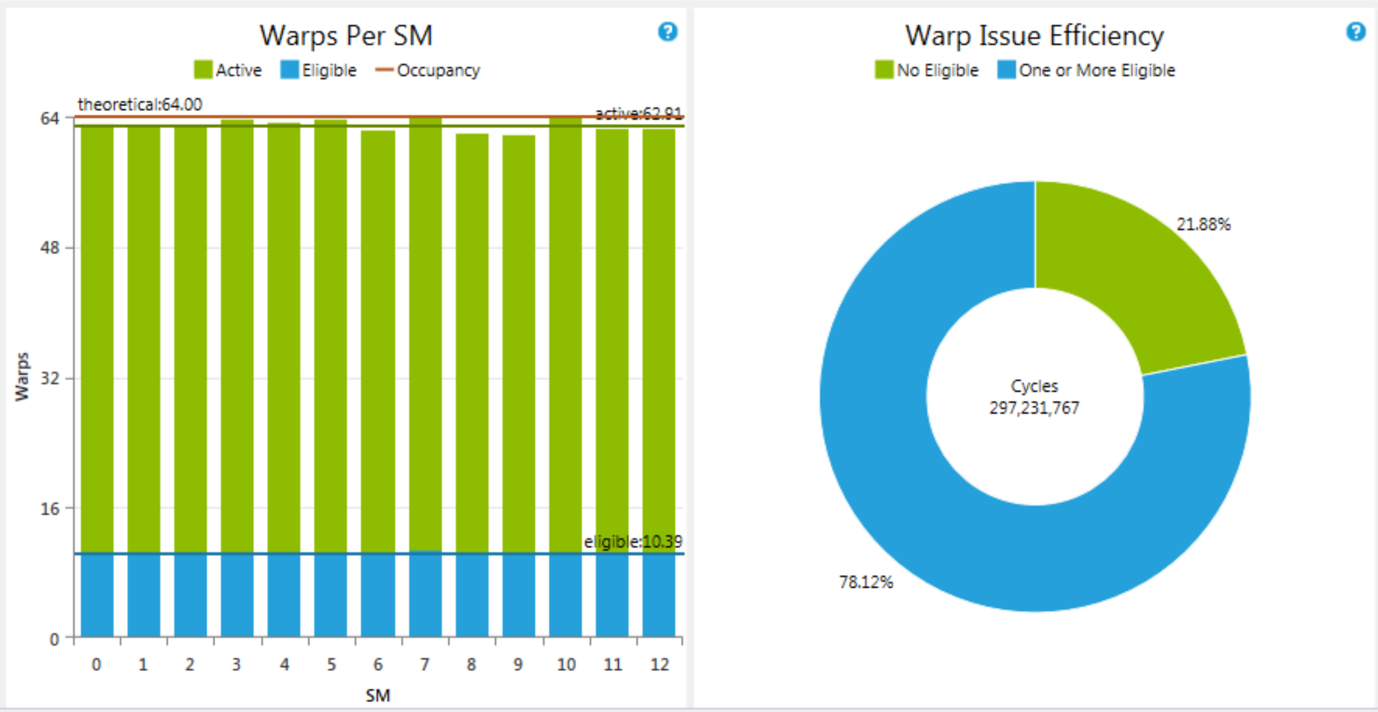
- Random data (uniform distribution), much slower to converge

	Tesla K20c (Time)	Passes
128	0.247s	68
256	2.498s	117

- For N = 256,
  - ½ of the cubes converge in 8 iterations
  - ¾ of the cubes converge in 11 iterations

THROUGHPUT LIMITED

- 78% issue efficiency (with high DRAM BW ~ 70% of peak)



GPU TECHNOLOGY CONFERENCE

THANK YOU

JOIN THE CONVERSATION

#GTC15