Real-time image segmentation for Homeland Security exploiting Hyper-Q concurrency

Fanny Nina-Paravecino
David Kaeli
NU-MGH CUDA Research Center
Dept. of Electrical and Computer Engineering
Northeastern University
Boston, MA

GPU TECHNOLOGY CONFERENCE

Northeastern

Homeland Security

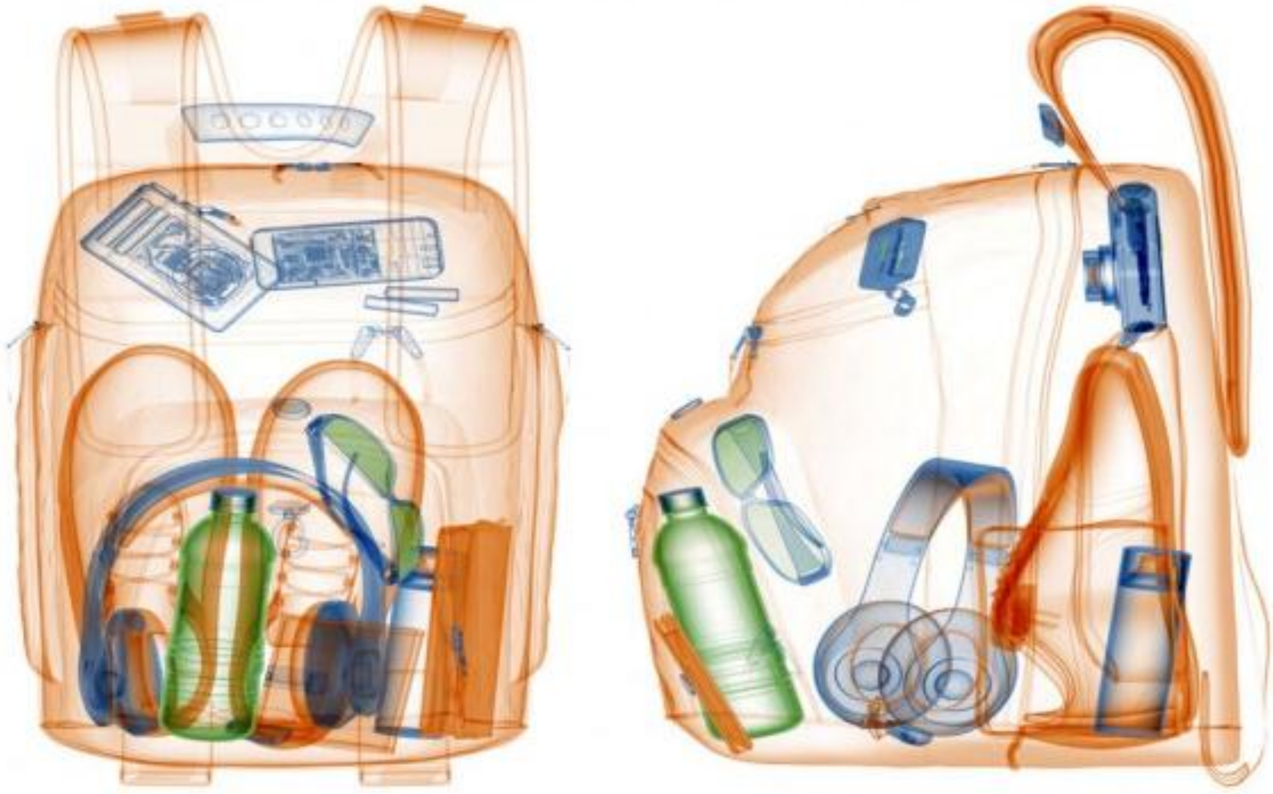


Image Credit: Thinkstock.com

Homeland Security

Alert!

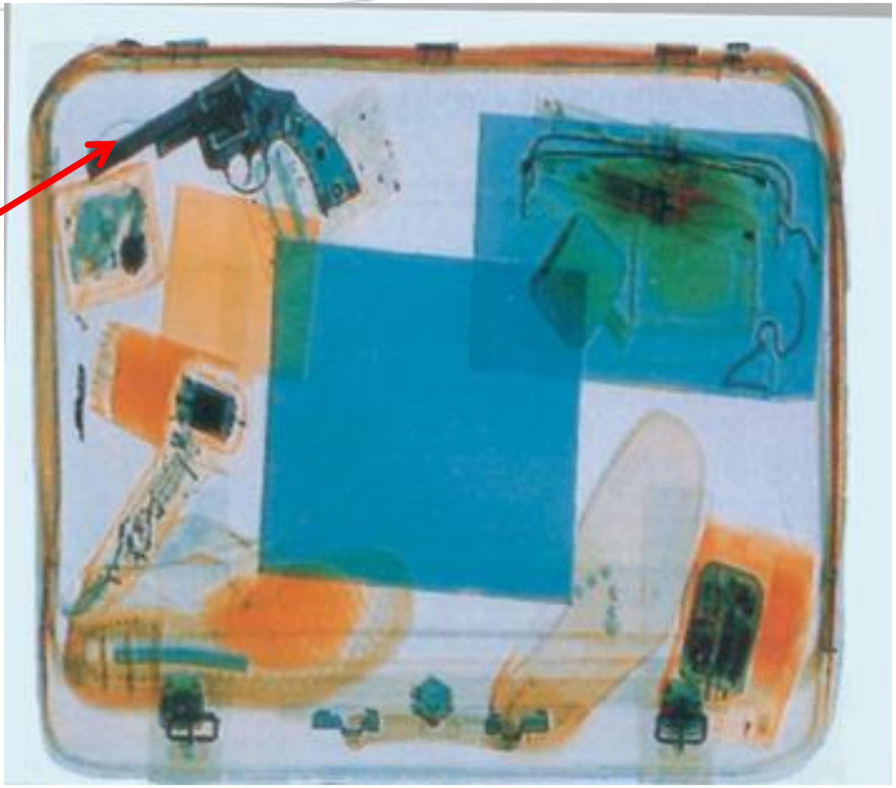
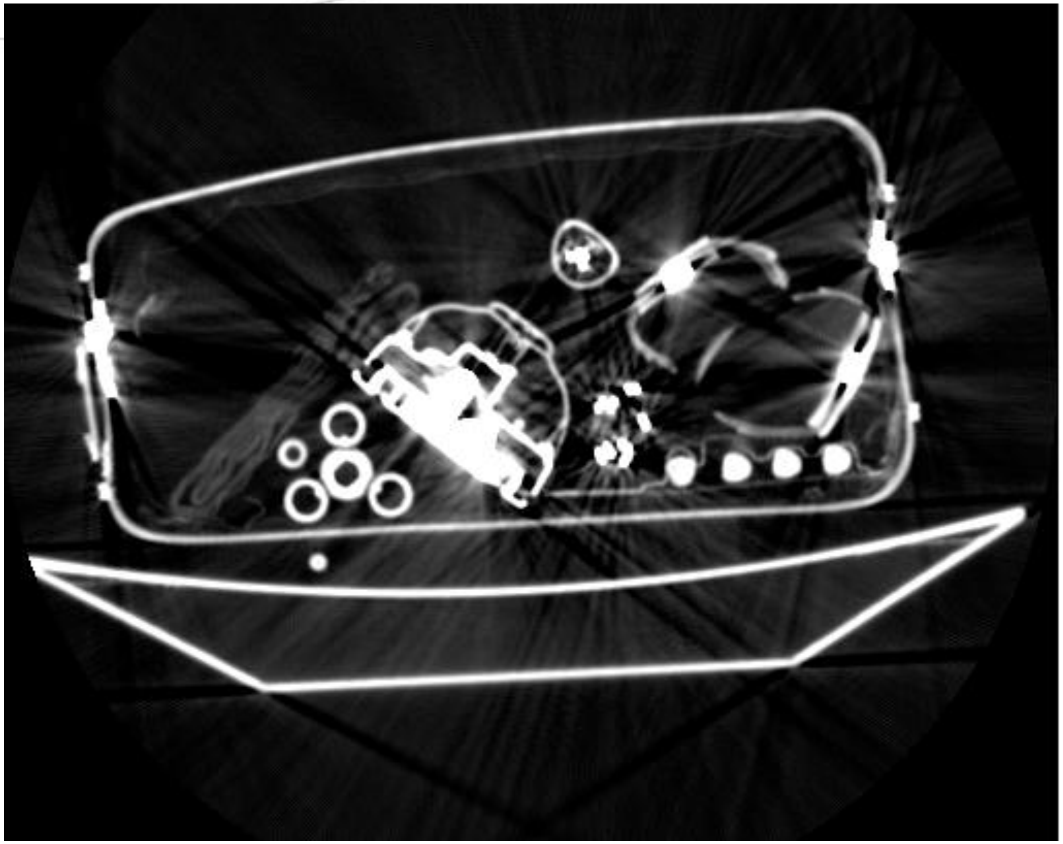


Image Credit: snallabolaget.com

Homeland Security

Constraints of the input data:

- Noise
- Hundreds of frames per objects



Homeland Security

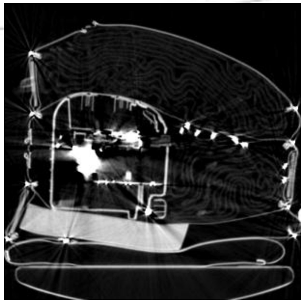
- One key application for Homeland Security is the need to perform high quality luggage inspection at airports
- This task becomes challenging since it involves the following constraints :
  - Near real-time response needed
  - Very high accuracy needed
- We will explore using CUDA 6.5 and new hardware features to address these needs in this important application

Outline for this presentation

- Background on the imaging analysis problem
- Connected Component Analysis
- Performance optimization
- NVIDIA's Hyper-Q
- Performance results
- Conclusion and future work

Homeland Security

One Frame DICOM Image



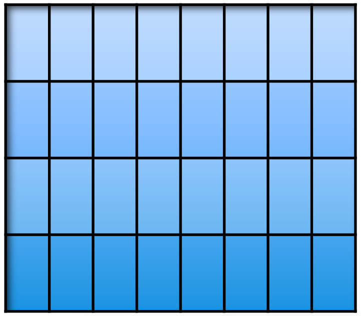
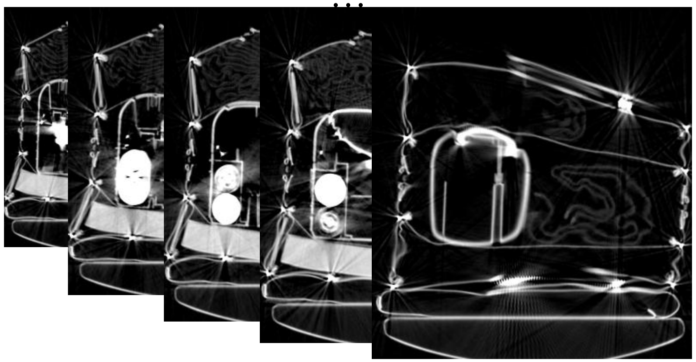
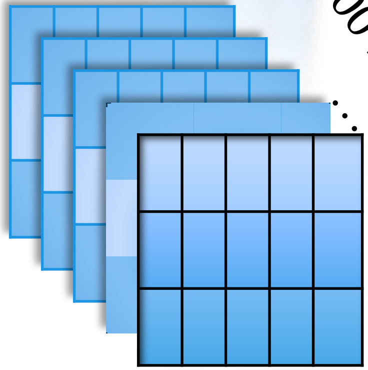


Image dimensions
512 x512

~700 Images

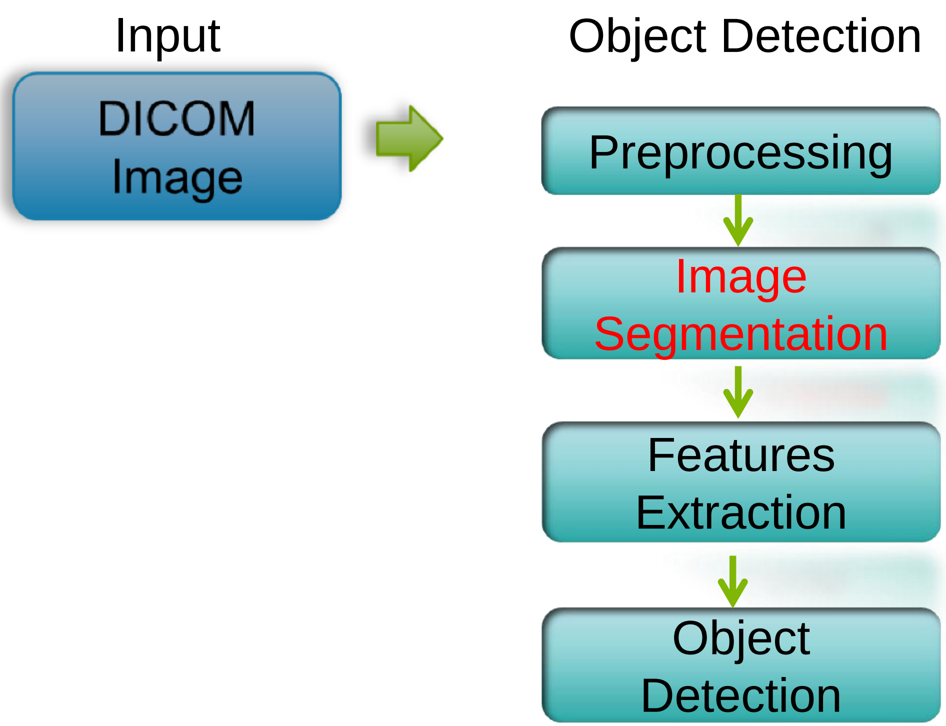
Multiple Frames





~700 matrices

Object Detection Pipeline



Homeland Security

- *Image Segmentation* plays a key role in the compute pipeline when performing object detection.
- Multiple algorithms:
  - Graph-based image segmentation [Fenzenswalb04]
  - Level Set [Shi05]
  - Spectral Clustering [Zelnik-Manor04]
  - Connected Component Labeling [Zhao10]

Outline for this presentation

- Background on the imaging analysis problem
- Connected Component Analysis
- Performance optimization
- NVIDIA's Hyper-Q
- Performance results
- Conclusion and future work

Connected Component Labeling

- Connected component labeling is a good fit based on the constraints of the environment
- Connected Component Labeling identifies neighboring segments possessing similar intensities
  - Potential for efficient segmentation
  - Provides high quality results

Connected Component Labeling

A lot of dependencies among neighbors!!!

1	1	2	2	2
1	2	3	3	3
2	2	3	3	4
2	3	4	4	4
2	3	2	2	2

Connected Component Labeling →

1	1	2	2	2
1	2	3	3	3
2	2	3	3	4
2	3	4	4	4
2	3	2	2	2

7 segments

Despite there are four different intensities. Groups pixels by location, and intensity

Outline for this presentation

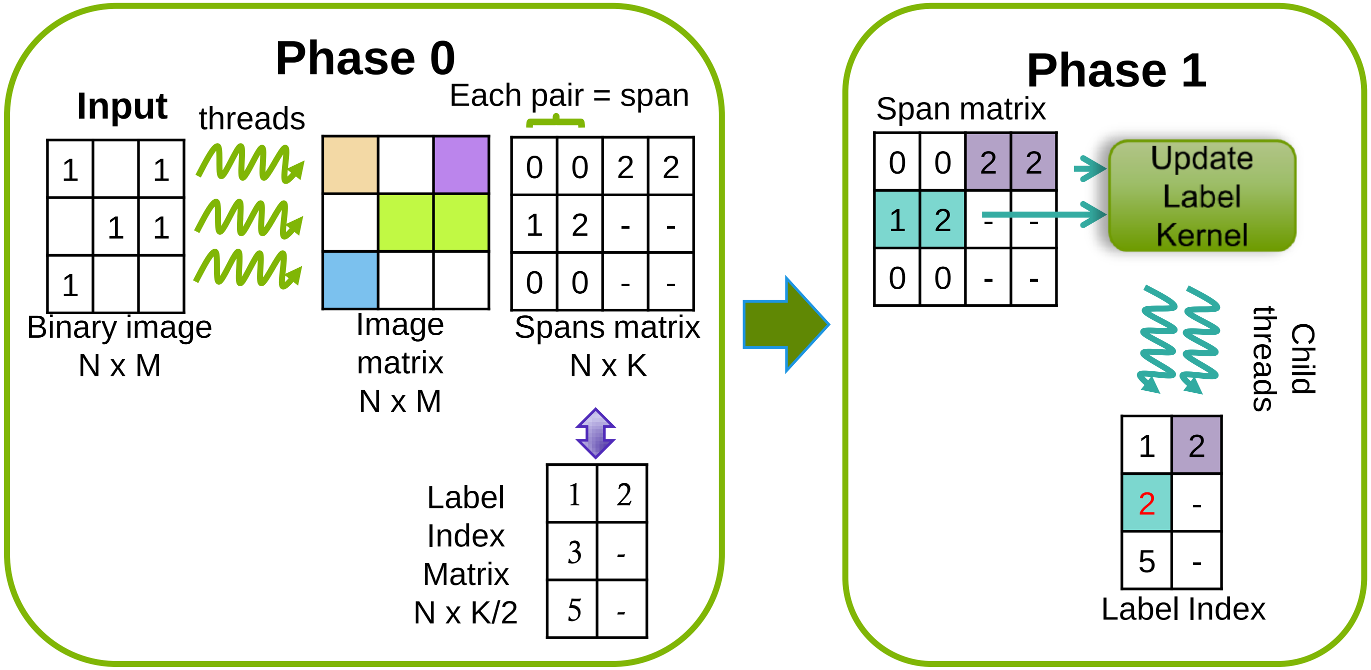
- Background on the imaging analysis problem
- Connected Component Analysis
- Performance optimization
- NVIDIA's Hyper-Q
- Performance results
- Conclusion and future work

How can we improve the performance of CCL?

- Exploit inherent parallelism!
- Dependencies among neighbors?
  - Stripe-based Connected Component Labeling [Zhao10]
  - Re-structure of the storage labeling
- Merge Strip-based approach?
  - Exploit CUDA's ***Dynamic Parallelism***
- Further optimizations
  - Explore the potential of using ***Hyper-Q***

Accelerated Connected Component Labeling

- Two phases:
  - Phase 0: Find Spans
  - Phase 1: Merge Spans

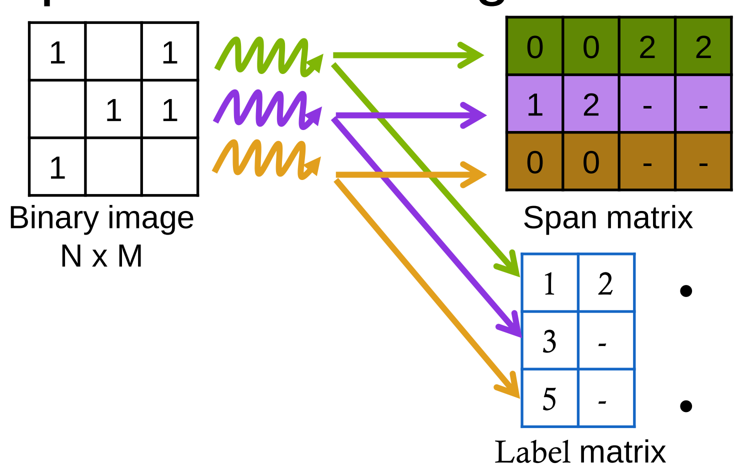


Phase 0: Find Spans

- Each span has two elements: (y_{start}, y_{end})

$$span_x = \{(y_{start}, y_{end}) \mid I_{(x, y_{start})} = I_{(x, y_{start+1})} = \ldots = I_{(x, y_{end})}\}$$

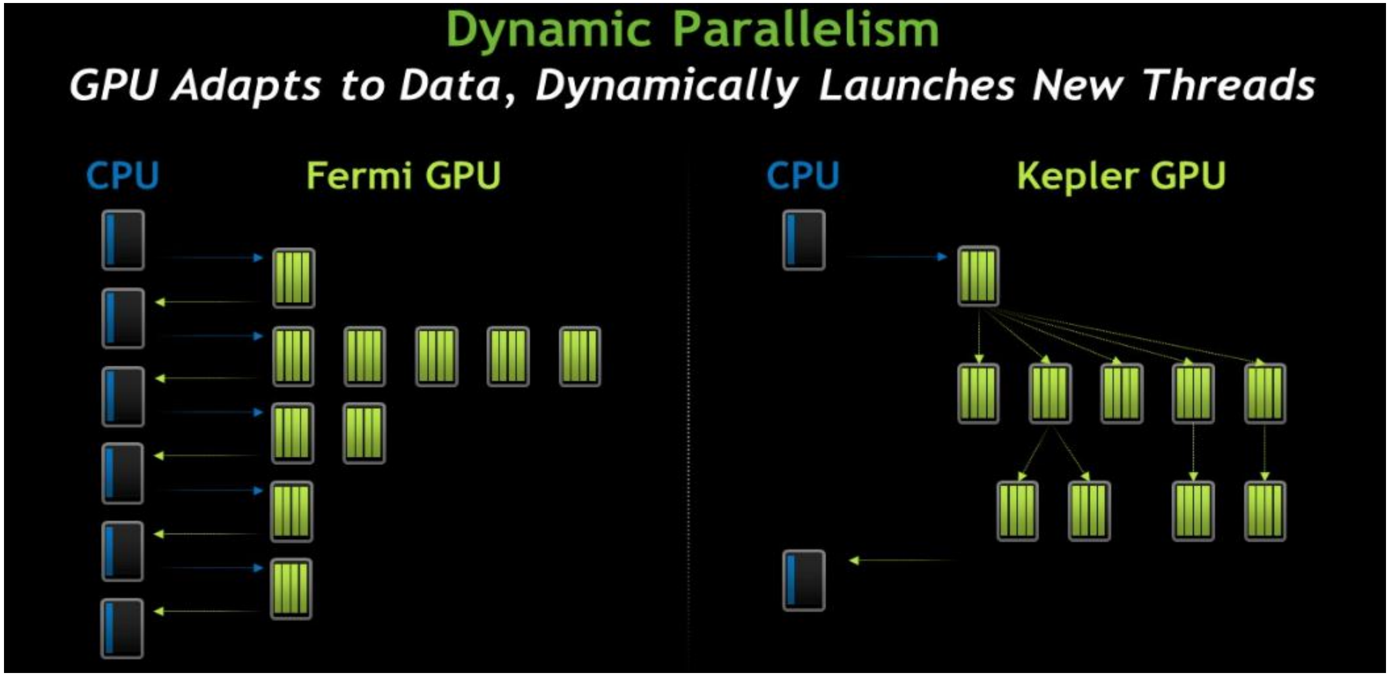
- A unique label is assigned immediately



- Reduced intermediate matrix of labels
- Half the size of the span matrix

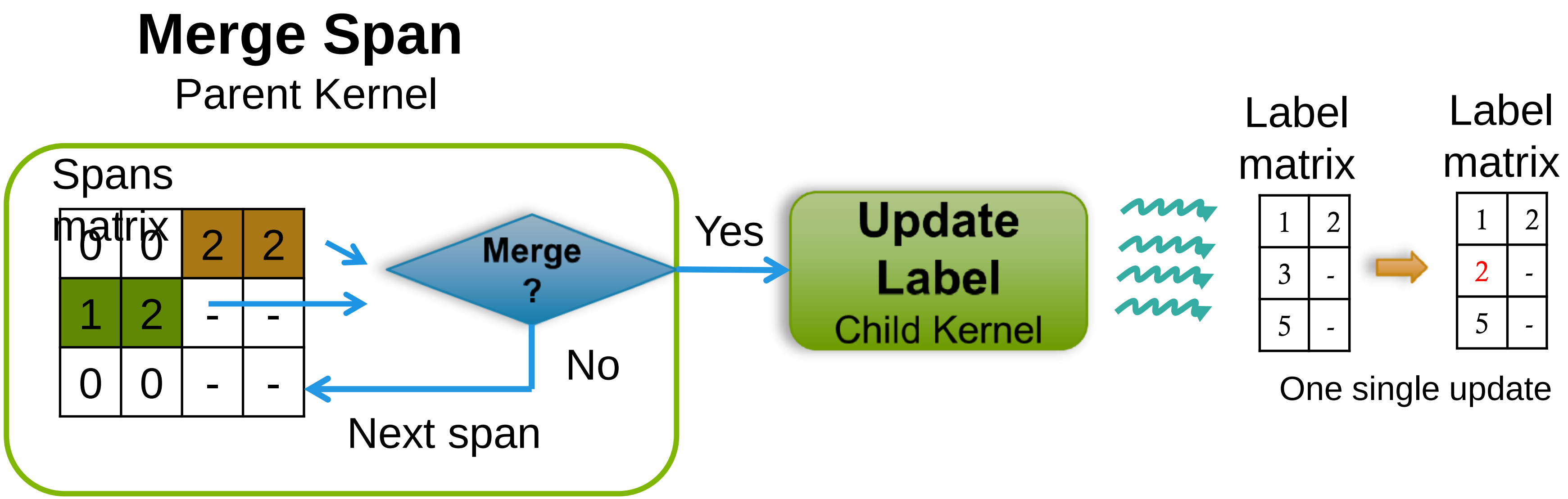
Dynamic Parallelism

- Kepler GK110 [Whitepaper NVIDIA's Next Generation CUDATM Compute Architecture: KeplerTM GK110]
  - Nested parallelism



Courtesy: NVIDIA

Phase 1: Merge Spans

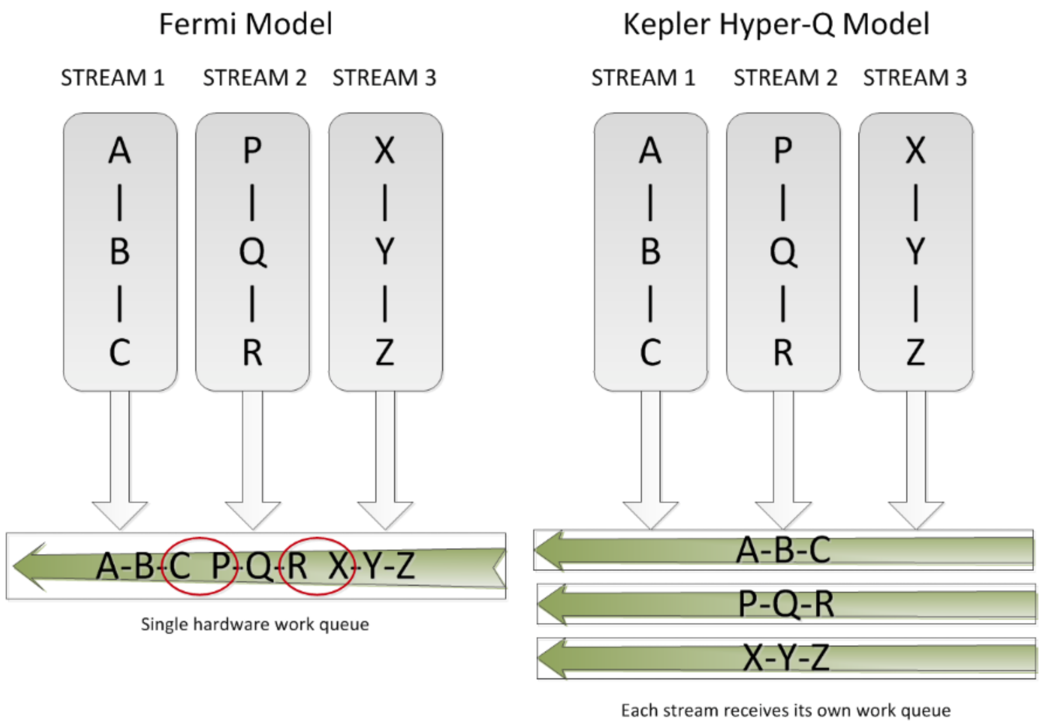


Outline for this presentation

- Background on the imaging analysis problem
- Connected Component Analysis
- Performance optimization
- NVIDIA's Hyper-Q
- Performance results
- Conclusion and future work

Hyper-Q

- Kepler: Hyper-Q working with CUDA streams [Whitepaper NVIDIA's Next Generation CUDATM Compute Architecture: KeplerTM GK110]



Courtesy: NVIDIA

When should we use Hyper-Q?

- Identify kernels that have low of the device
- Identify applications that can allow for concurrent kernel execution
- Two tasks:
  - Analyze the applications
  - Analyze the kernels

Outline for this presentation

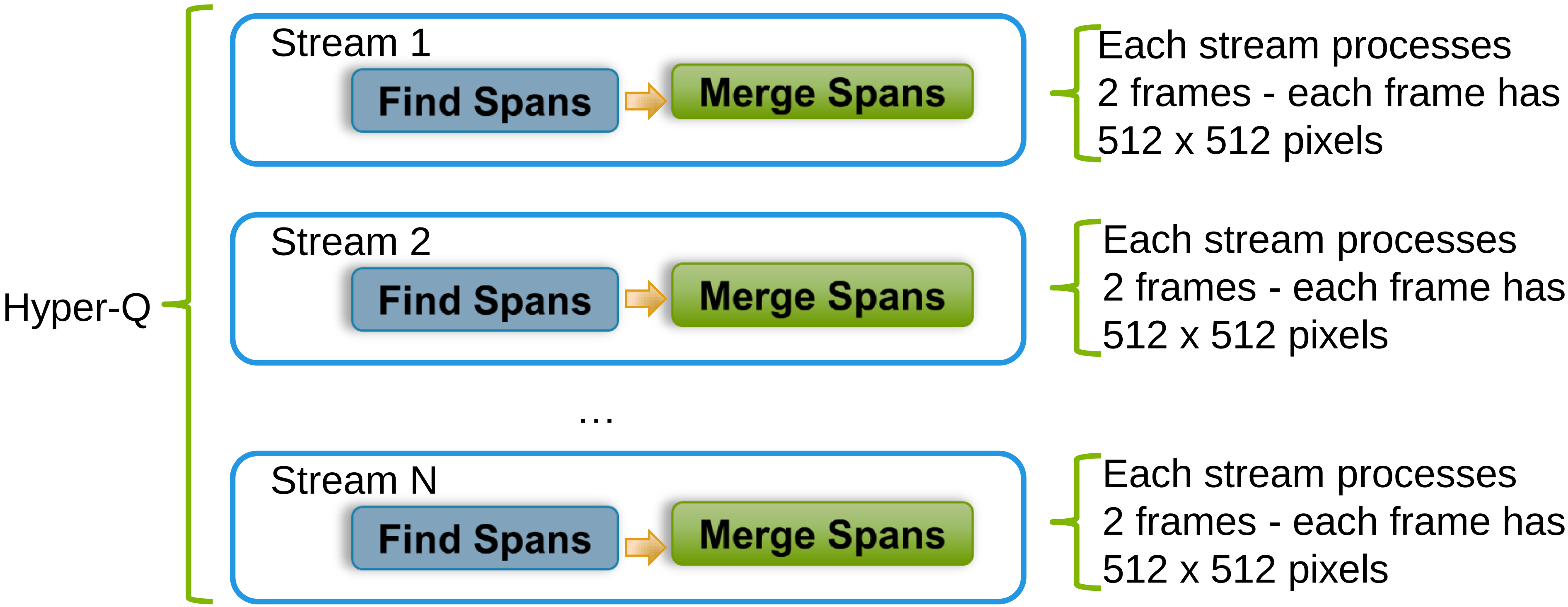
- Background on the imaging analysis problem
- Connected Component Analysis
- Performance optimization
- NVIDIA's Hyper-Q
- Performance results
- Conclusion and future work

Accelerated Connected Component Labeling

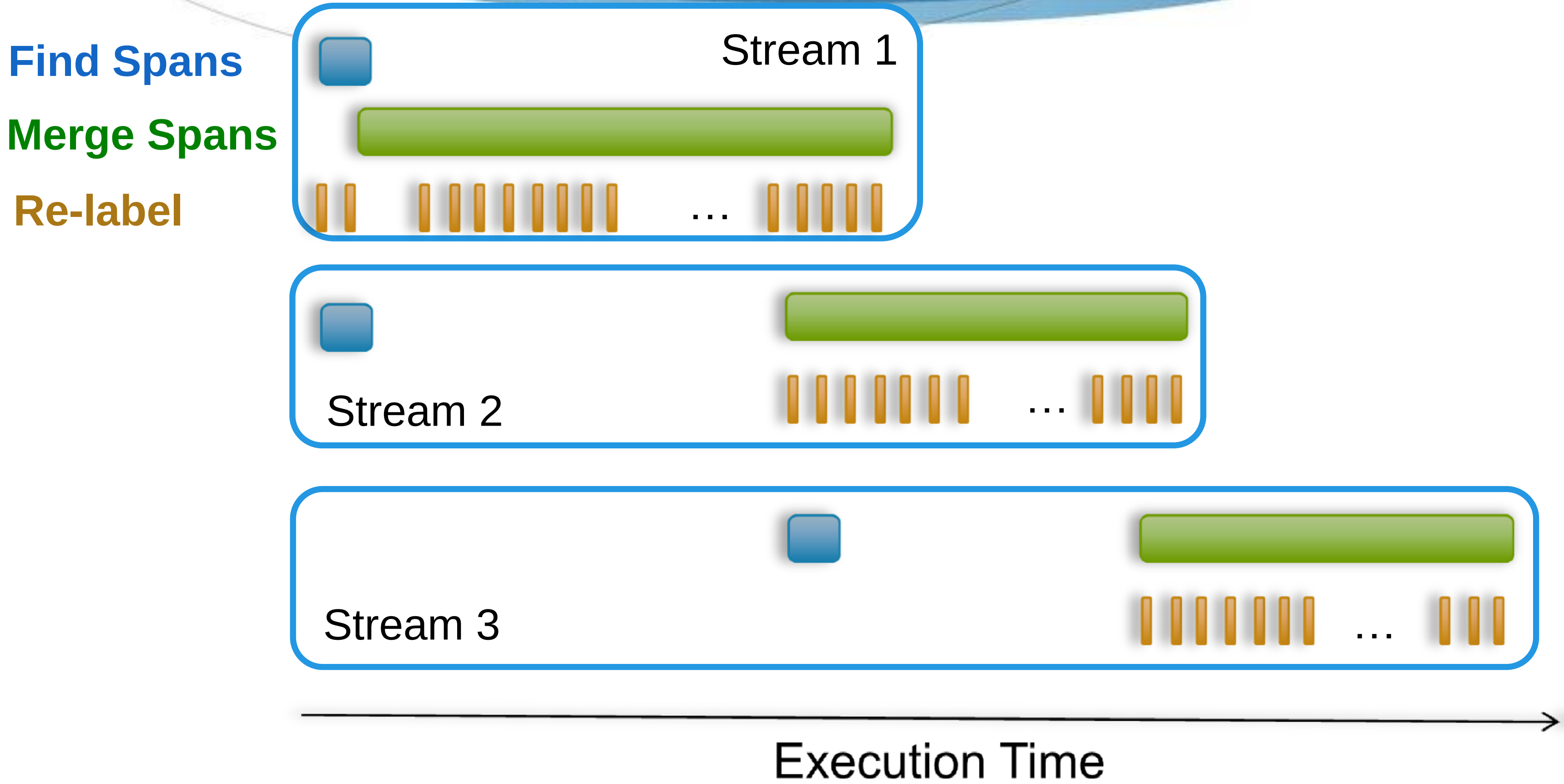
- Resources utilization per kernel
  - Find Spans:
    - SMX Activity: 27%
    - Occupancy: 0.11
  - Merge Spans
    - SMX Activity: 31%
    - Occupancy: 0.09

Accelerated Connected Component Labeling

- Exploiting Hyper-Q



Concurrent kernel execution



Performance Results

- Speedup of a stream-based ACCL run on CUDA 6.5 vs. OpenMP with 8 threads on an Intel Core i7-3770K

# Streams	# Frames	OpenMP CCL (s)	ACCL(s)	Speedup
4	8	2.72	1.35	2.01x
8	16	10.79	2.73	3.94x
16	32	42.92	5.43	7.91x
32	64	171.18	10.79	15.32x
64	128	1020.00	21.56	47.32x

Outline for this presentation

- Background on the imaging analysis problem
- Connected Component Analysis
- Performance optimization
- NVIDIA's Hyper-Q
- Performance results
- Conclusion and future work

Conclusion

- Improved performance of image segmentation task for baggage scanning problem
- Exploited NVIDIA's Hyper-Q feature to accelerate Connected Component Labeling
- Compared an OpenMP CCL implementation with our ACCL implementation
  - Our algorithm scales well as long as we increase the number of streams
- Kernels with low occupancy are the best fit to use Hyper-Q

Future work

- Combine Hyper-Q with MPI to exploit multiple grains of parallelism using multiple GPU nodes
- Evaluate additional image segmentation algorithms that address the constraints of baggage scanning

THANK YOU

- **Questions?**
- fninaparavecino@ece.neu.edu

analogic

NVIDIA

Northeastern