An Evaluation of the Potential of GPUs to Accelerate Tracking Algorithms for the ATLAS Trigger

John Baines, Timothy Bristow, Dmitry Emeliyanov, **Jacob Howard**, Sami Kama, Matthew Robson, Andrew Washbrook, Ben Wynne

GPU Computing in High Energy Physics 2014
11 September 2014

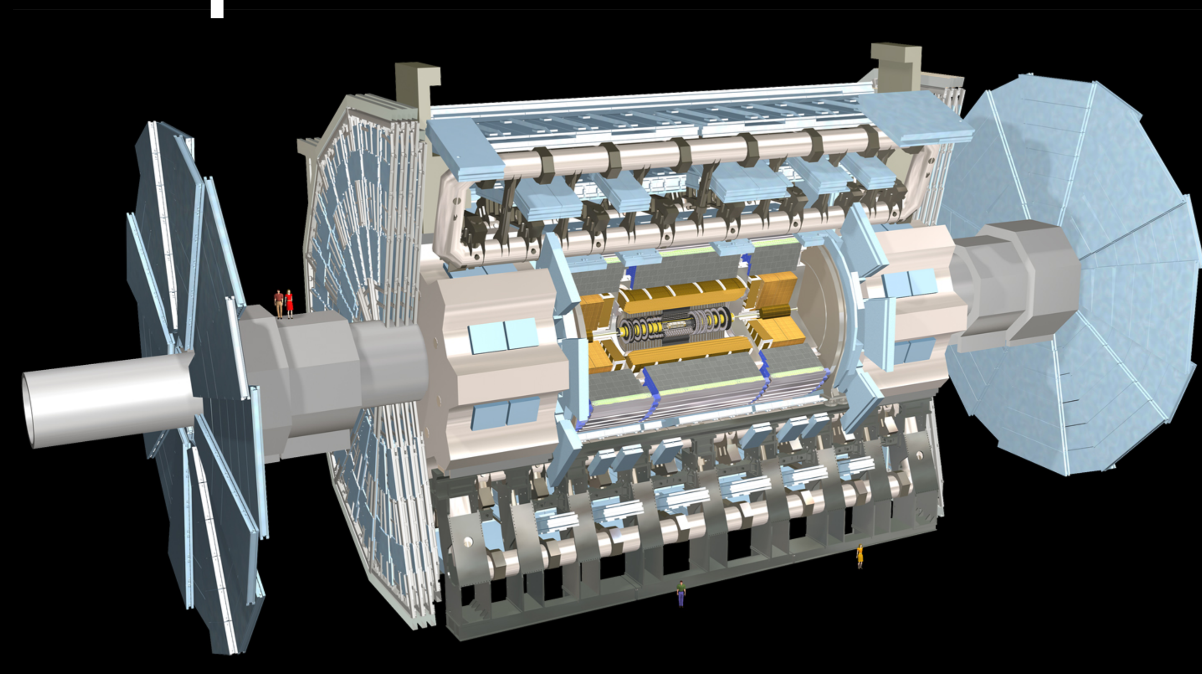
ATLAS EXPERIMENT
Science & Technology Facilities Council
Rutherford Appleton Laboratory
UNIVERSITY OF OXFORD
THE UNIVERSITY of EDINBURGH
SMU

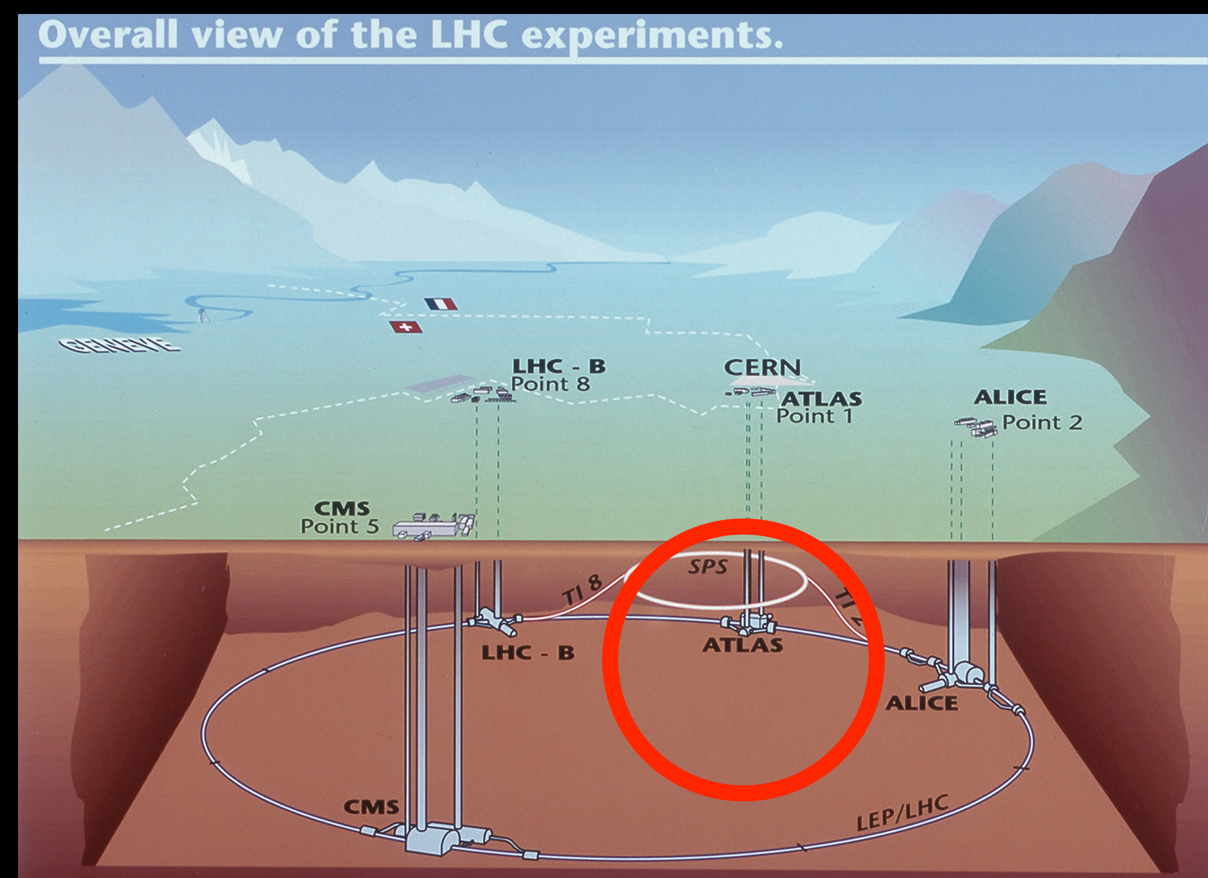
Outline

- The ATLAS experiment, detector, and trigger
- Trigger data preparation and tracking algorithms
- Motivation for using GPUs
- Parallelized algorithms
- Performance results
- Porting complexities
- Conclusion and outlook

The ATLAS Experiment

- One of two general-purpose particle detectors at the **Large Hadron Collider** (LHC)
- Played an important role in the 2012 discovery of a particle consistent with the Standard Model Higgs boson
- Contains an assortment of trackers, calorimeters, and muon detectors - all nested in an onion-like fashion
- Will commence second run of operation in early 2015 at higher energy and luminosity





Images courtesy of CERN and the ATLAS Experiment

The Inner Tracker

- Innermost component of the detector is a silicon-based tracker designed to map the trajectories of particles emanating from collisions
- Consists of nested layers of silicon detectors
    - Innermost layers are **pixel detectors**
    - Outermost layers are strip detectors (**SCT**)
- Sits inside a solenoid magnet, causing the tracks of charged particles to curve inversely proportional to their momentum

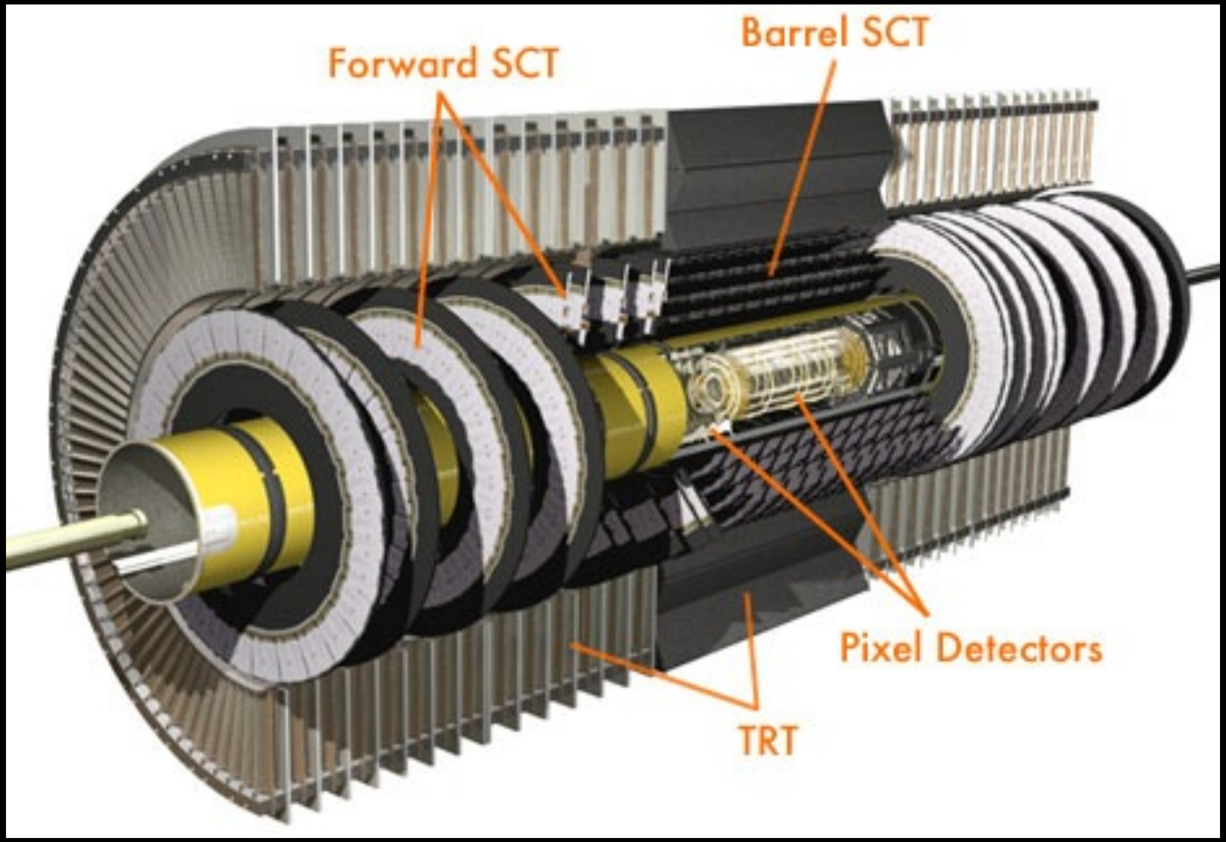
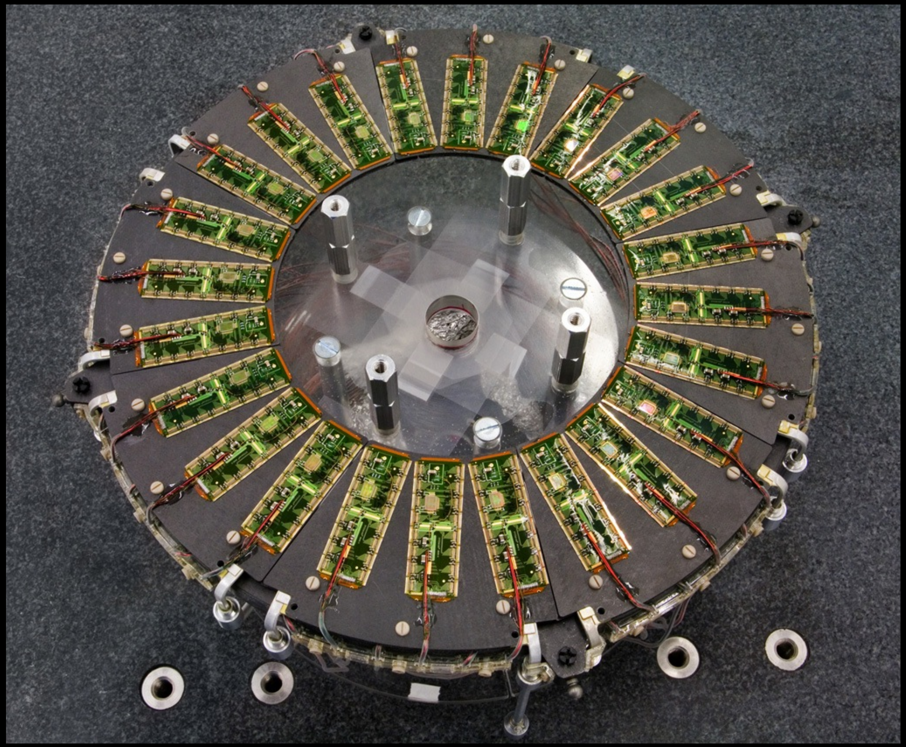
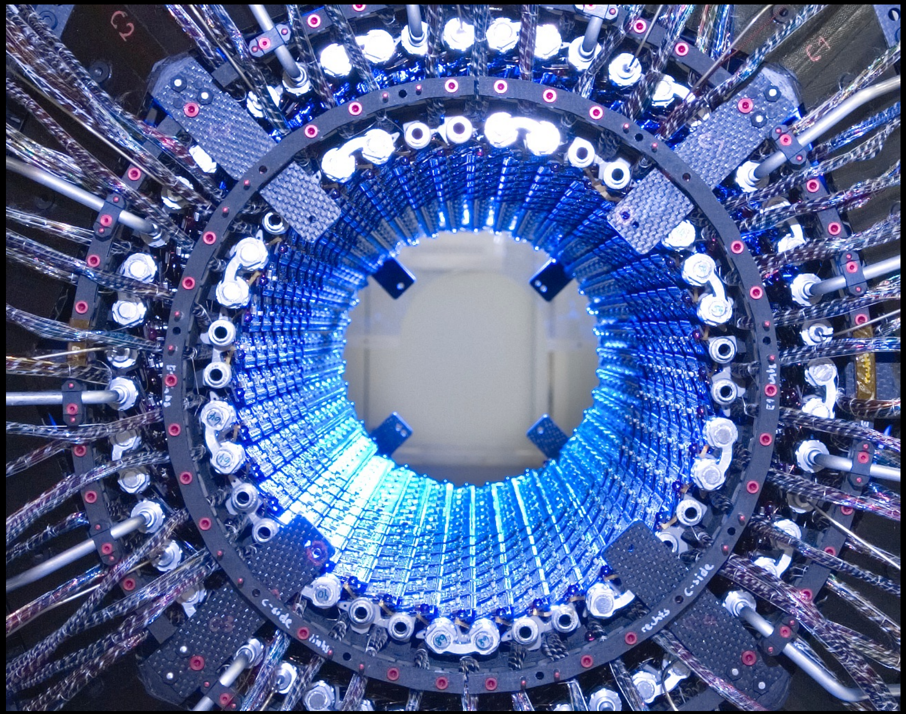


Image courtesy of CERN and the ATLAS Experiment

The Pixel Detector

- Comparable to a gigantic digital camera (operating at 40 MHz!)
- Composed of 1,744 **modules** in concentric layers and end-caps
- Each module contains approximately 46,000 pixels - over 80,000,000 pixels!
- As charged particles fly though, they ionize the silicon and create a detectable charge distribution
- A single particle may activate several adjacent pixels
- Innermost layer only 5 cm from the beam axis
- Present upgrade will add layer of 14 million pixels 3.2 cm from the beam axis

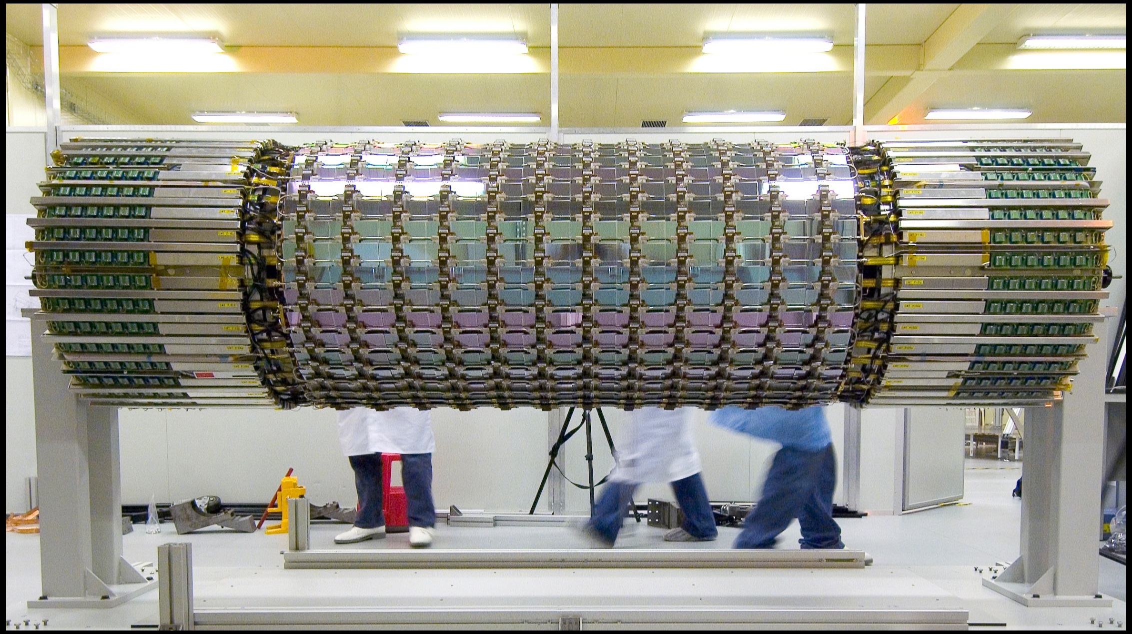


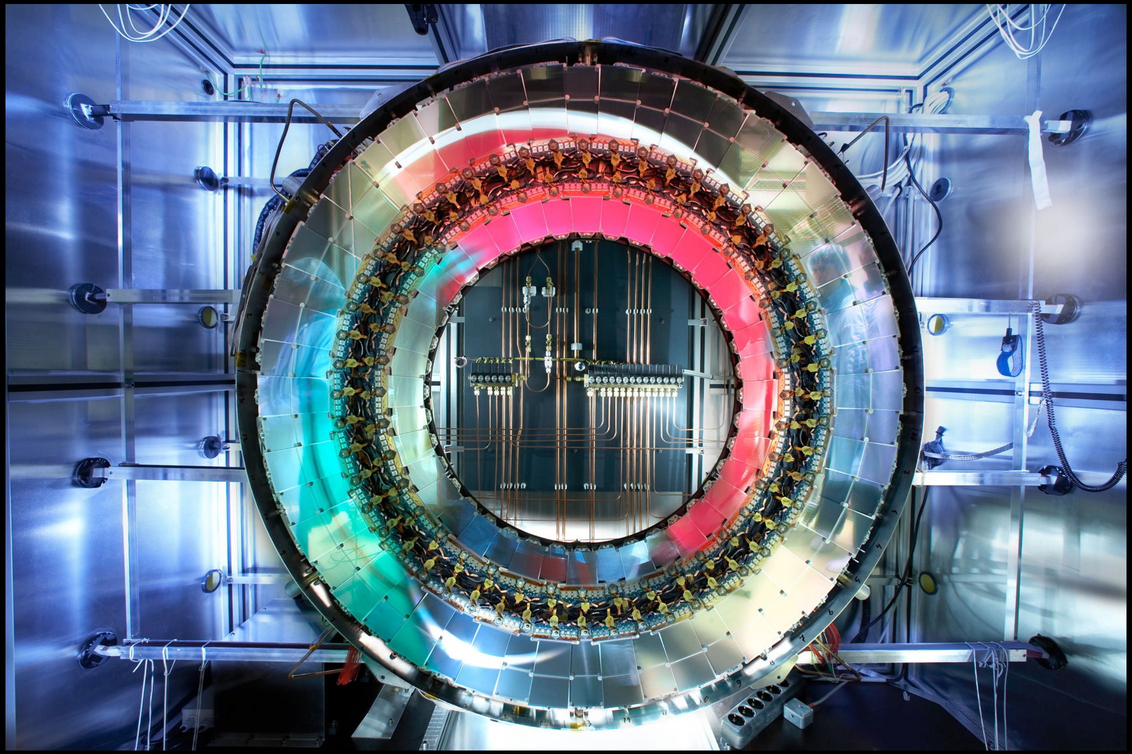


Images courtesy of CERN and the ATLAS Experiment

The SCT Detector

- Similar in purpose to the pixel detector, but farther from the beam axis
- Lower resolution requirements allow for a savings in readout bandwidth and manufacturing cost
- Composed of 8,176 modules in concentric layers and end-caps
- Each module consists of two layers of strips, with the layers at a small relative angle
- Charged particles will activate both layers of strips, and the angle between them can determine the position, with some ambiguity





Images courtesy of CERN and the ATLAS Experiment

The ATLAS Trigger

- Far too much data generated to store permanently or analyze
- Run I used a cascading three-tier **trigger** system to select events

Level	Type	Analysis	Input Rate	Output Rate	Execution Time
1	Hardware	Calorimeter and muon data	20 MHz	70 kHz	2.5 μs
2	Software	Incorporate inner detector, use fast custom algorithms	70 kHz	6.5 kHz	90 ms
Event Filter	Software	Near-offline reconstruction	6.5 kHz	600 Hz	1 s

- Software tiers will be merged in Run II, with an input rate of ~100 kHz and processing time of ~250 ms

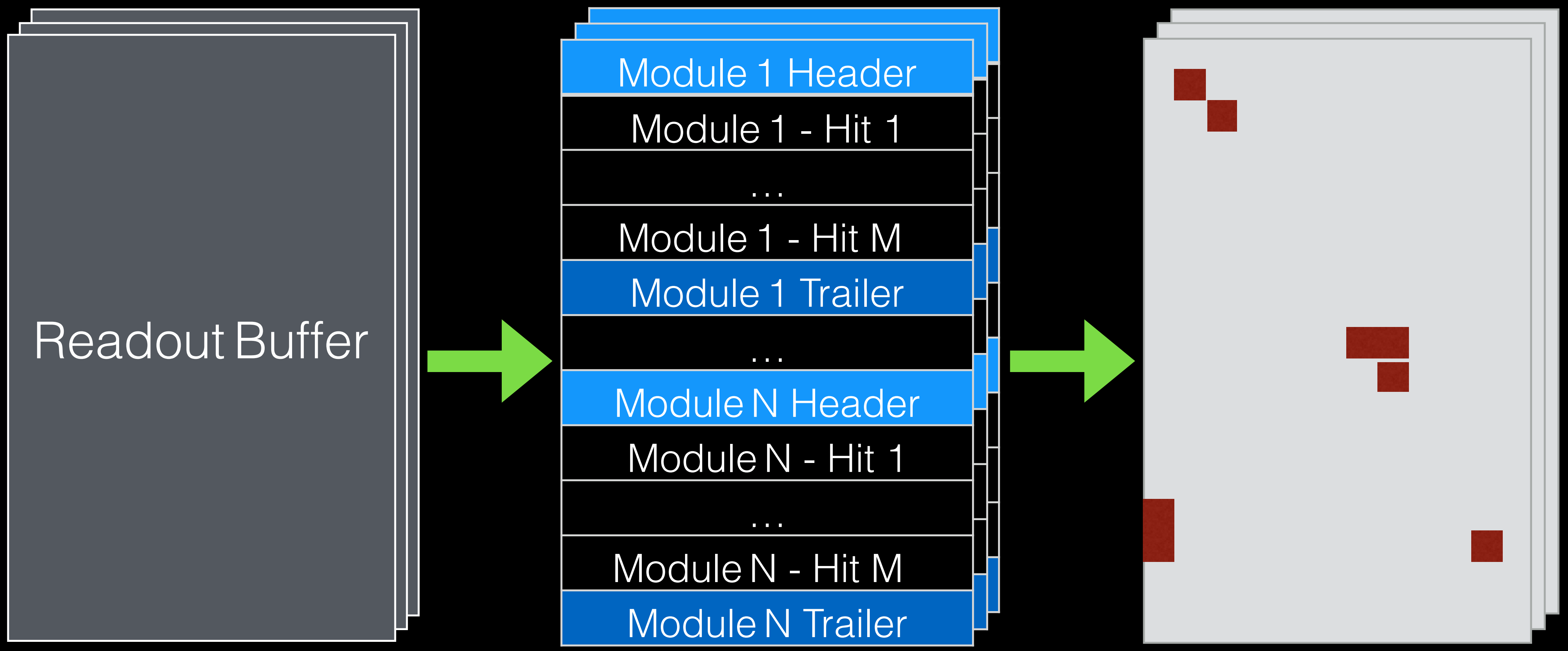
ATLAS Trigger Algorithms

- Trigger analysis in software triggers requires reconstruction of particle tracks
- The reconstruction is seeded by solid angle **Regions of Interests** identified by the Level 1 trigger
- Particle hits in pixel and SCT modules are encoded in a compact **bytestream** format and stored in **Readout Buffers**
- Reconstruction is a four-step process:

- Each of these steps requires significant processing time

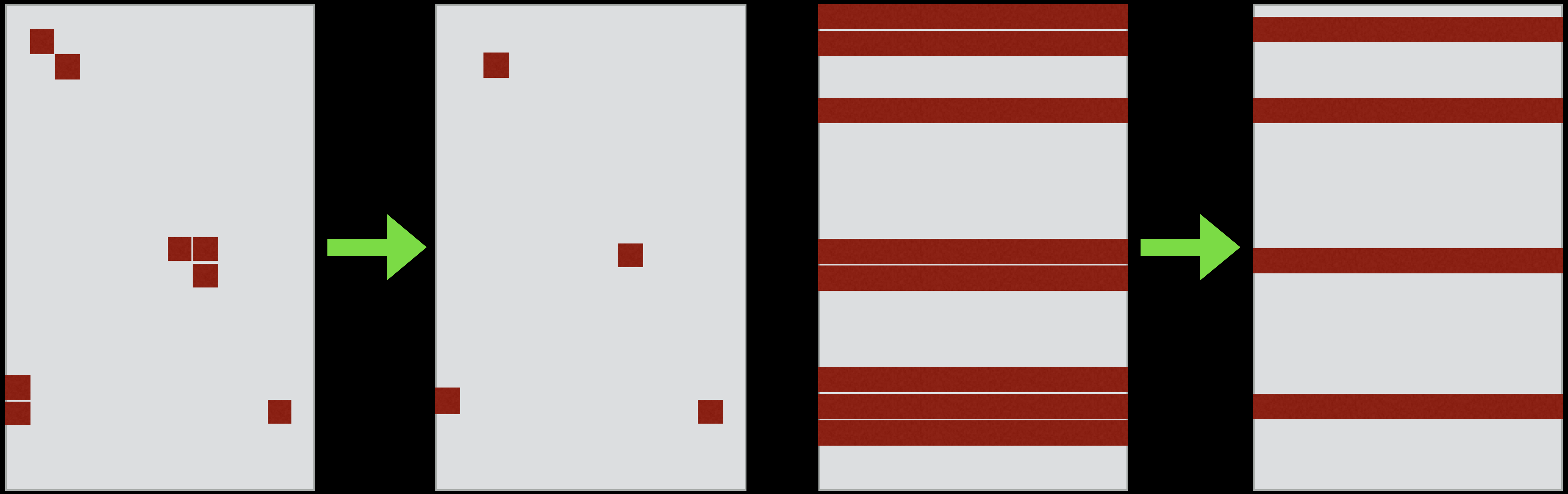
Bytestream Decoding

- Bytestream data is first retrieved by requesting it from the Readout Buffers via a network connection
- The bytestream consists of 32-bit/16-bit words for the pixel/SCT detectors
- The structure of the bytestream and context of its constituent words encode module identifiers and the hits belonging to them



Hit Clustering

- Multiple silicon cells activated by a single particle must be clustered together
- For pixel modules, this is done by checking hits for adjacency with known clusters, and merging clusters which are adjacent to the same hit
- For SCT modules, clustering is trivial since adjacency need only be determined in one dimension
- Clusters are then converted to **spacepoints** by translating/rotating to match the physical module position/orientation



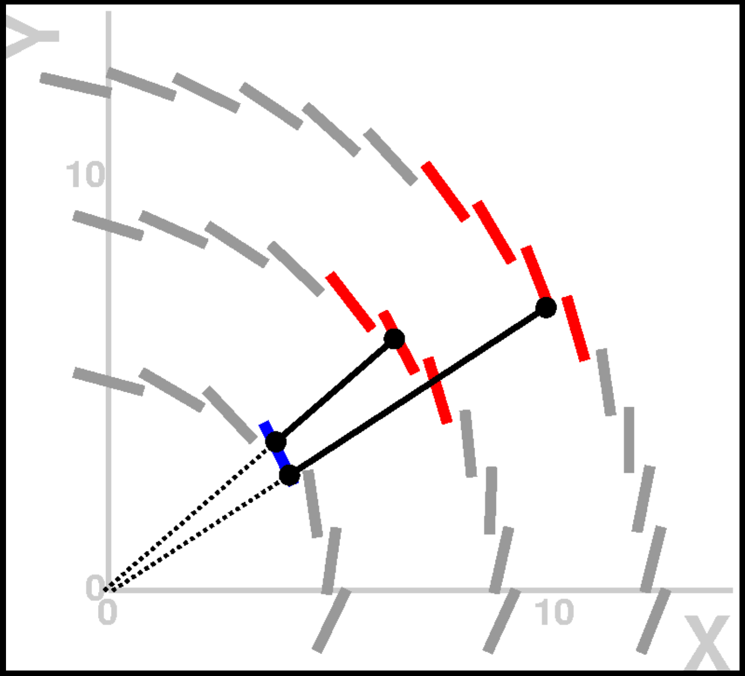
Pixel Clustering

SCT Clustering

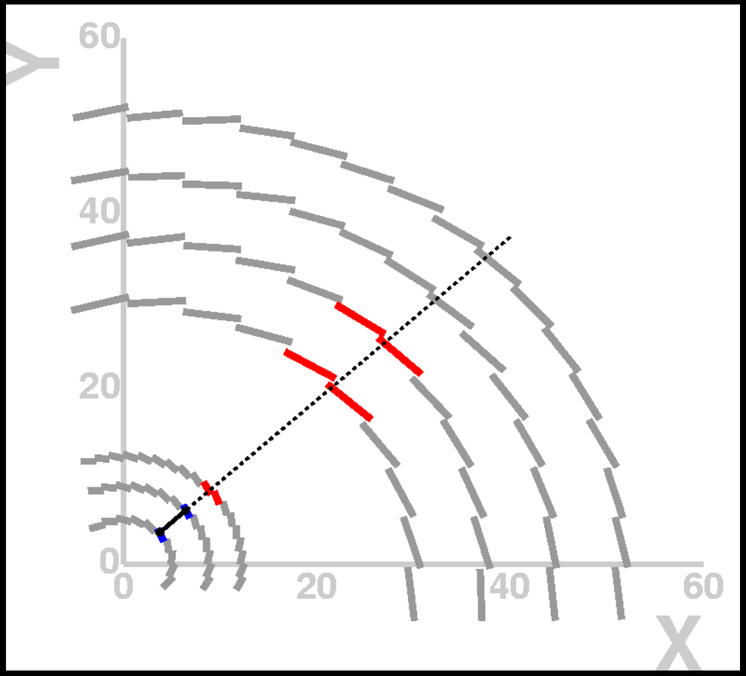
Track Formation and Clone Removal

- Track seeds are first formed by combining points in inner silicon layers

- Seeds are extended to include points in outer silicon layers



Track seed formation



Track seed extension

- Multiple **clone** tracks may be identified with the same outer hits and different seeds - they must be identified and then merged/removed

Motivation for Using GPUs

- Data preparation and tracking are some of the most computationally intensive trigger steps (50-70% of processing time)
- An increase in the instantaneous luminosity of LHC proton beams will lead to a proportional increase in events per proton-proton bunch crossing, increasing hit occupancy
- Combinatorial nature of the track reconstruction will lead to a large increase in serial processing time
- GPUs offer massive parallelization potential over CPUs
- Chose CUDA due to maturity, support, and ease of development

Year	Peak Instantaneous Lumionsity
2010	2.1×10^{32} $cm^{-2}s^{-1}$
2011	3.65×10^{33} $cm^{-2}s^{-1}$
2012	7.73×10^{33} $cm^{-2}s^{-1}$
2015	1.6×10^{34} $cm^{-2}s^{-1}$
HL-LHC	$5\text{-}7 \times 10^{34}$ $cm^{-2}s^{-1}$

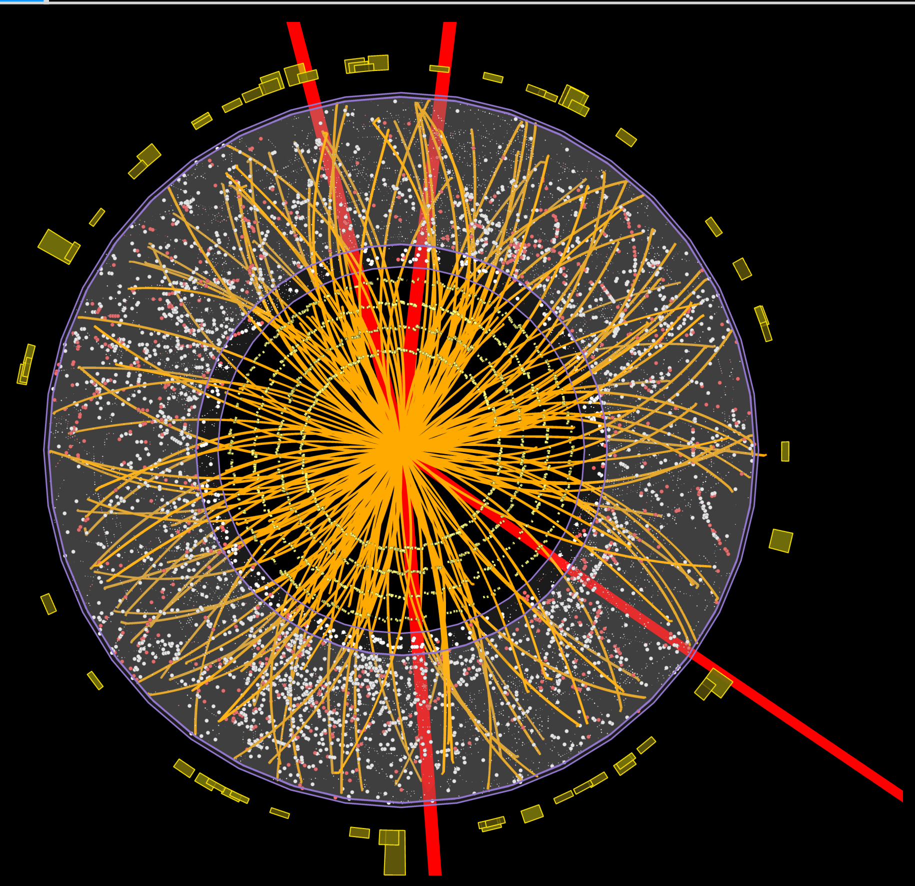


Image courtesy of CERN and the ATLAS Experiment

Parallel Bytestream Decoding

- Bytestream fragments from different Readout Buffers are mapped to different thread blocks/streaming multiprocessors

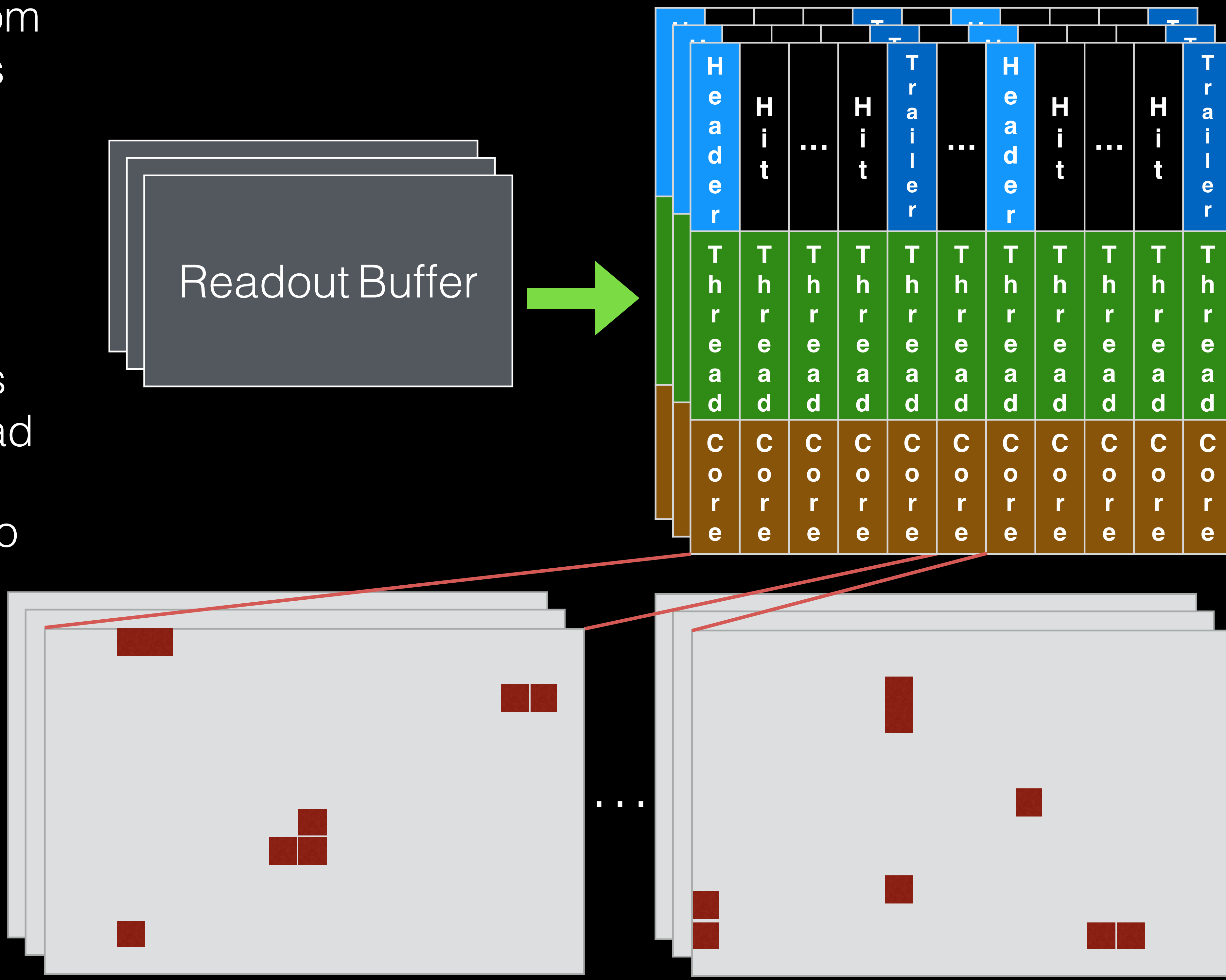
- Within fragments, each word in the bytestream is mapped to a single thread

- Each thread performs two operations:

    - Context detection

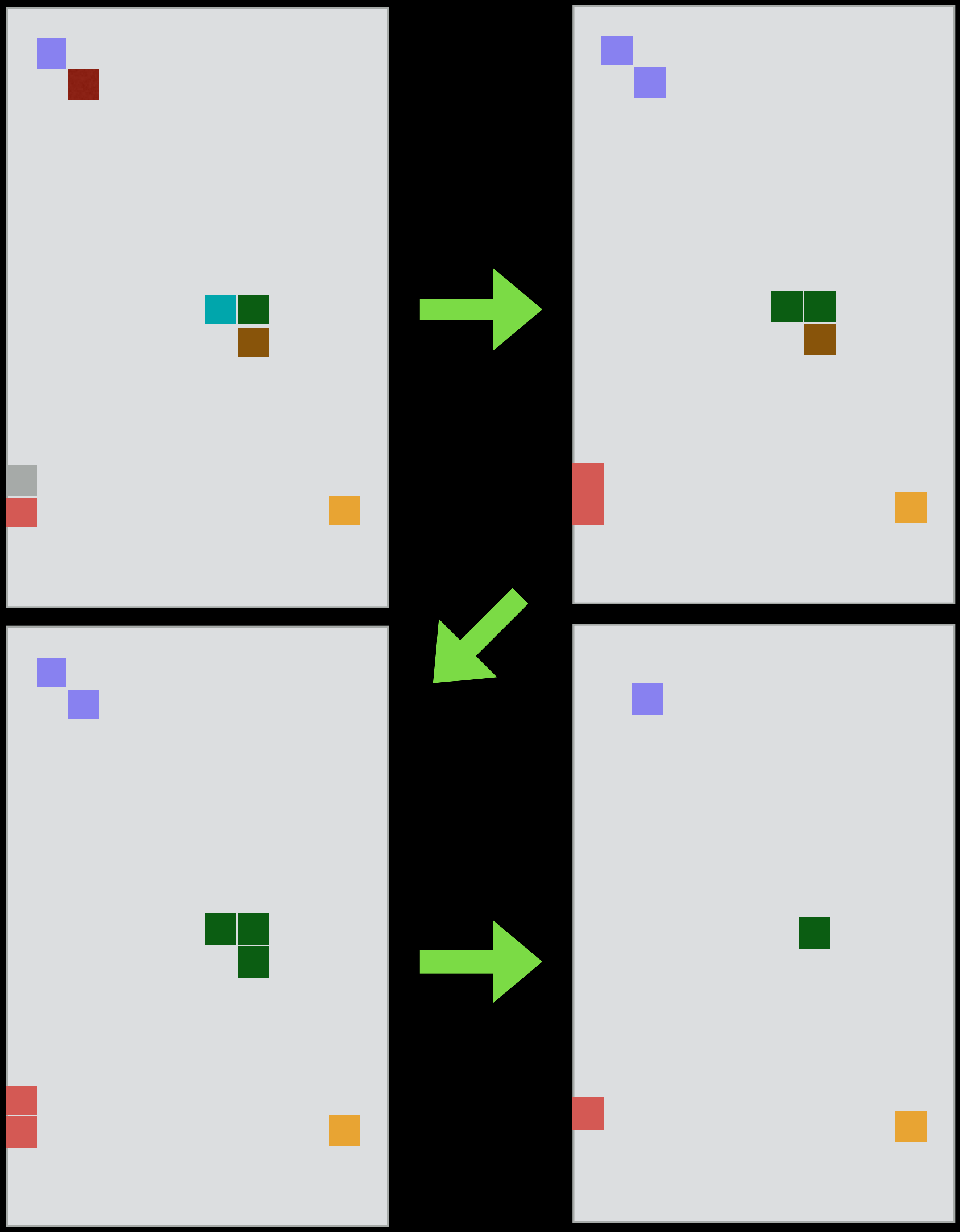
    - Value decoding

- Threads handle multiple words depending on largest thread block size



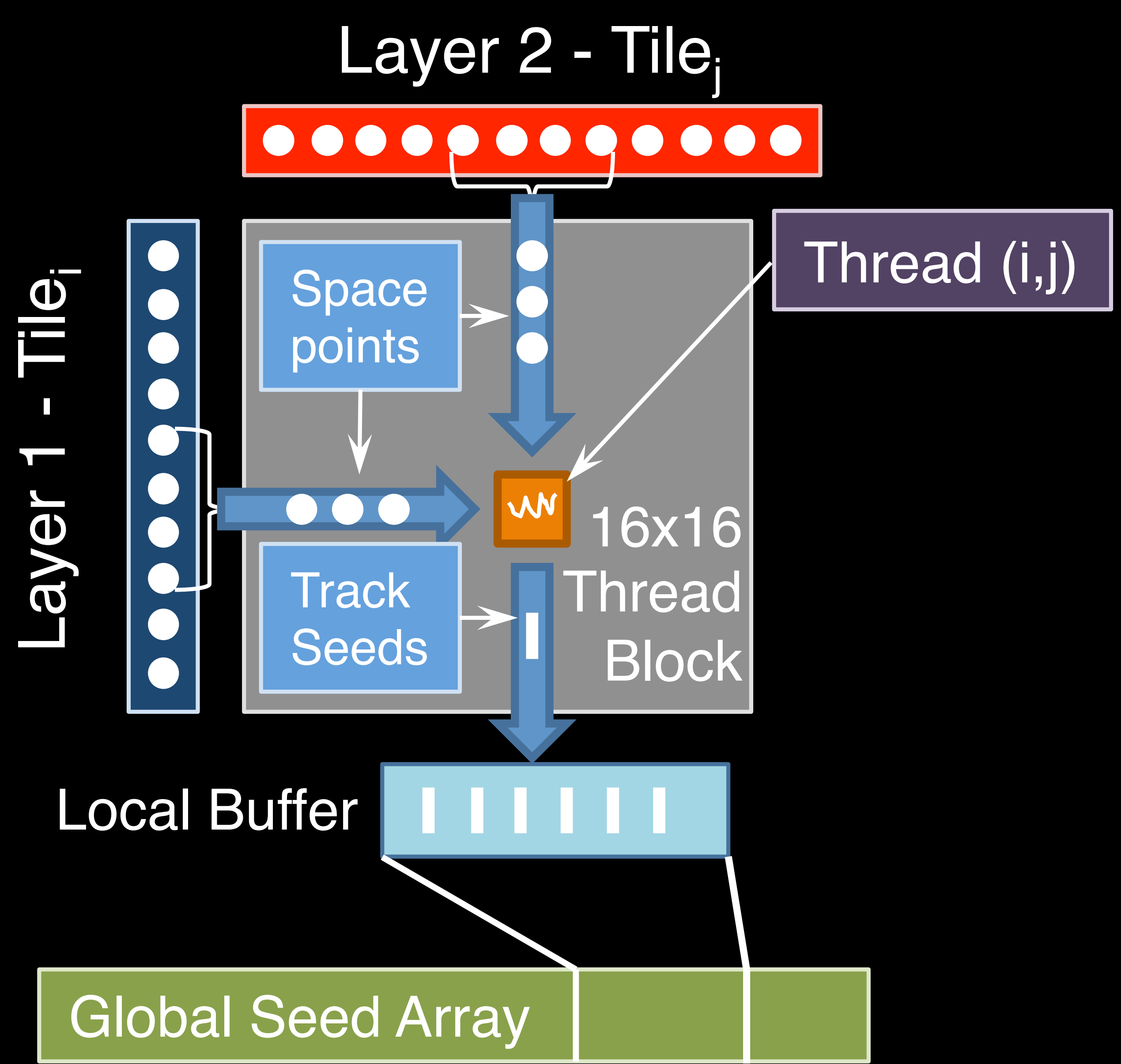
Parallel Hit Clustering

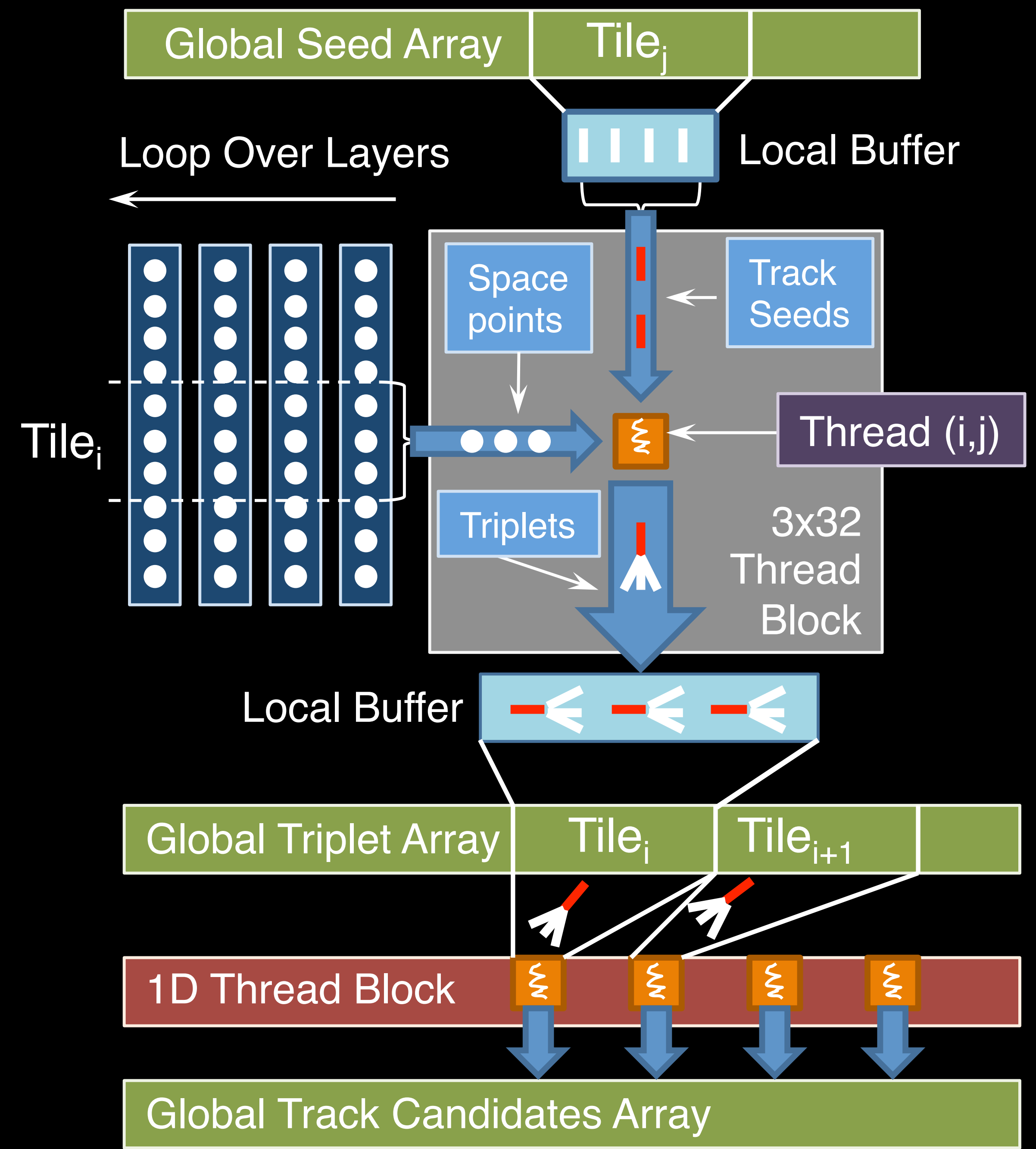
- A cellular automaton is used to iteratively combine hits into groups
- All hits are assigned an initial tag
- Each evolution sets the tag to the highest of those adjacent to it
- Clustering is complete when the automaton stops evolving



Parallel Track Formation

Track seeds are first identified by a 2-dimensional thread block

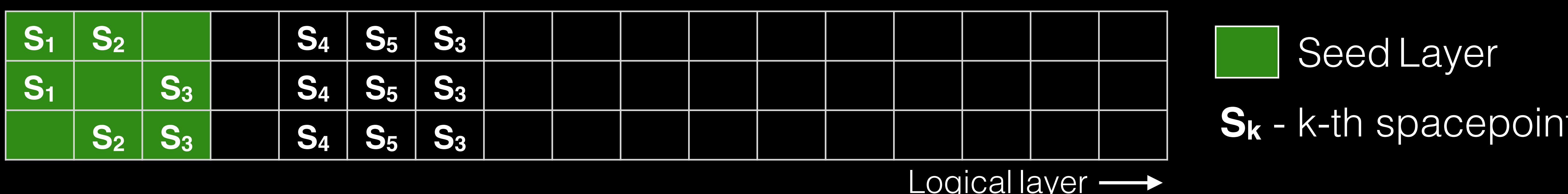




Track seeds are extended to outer layers and merged into track candidates

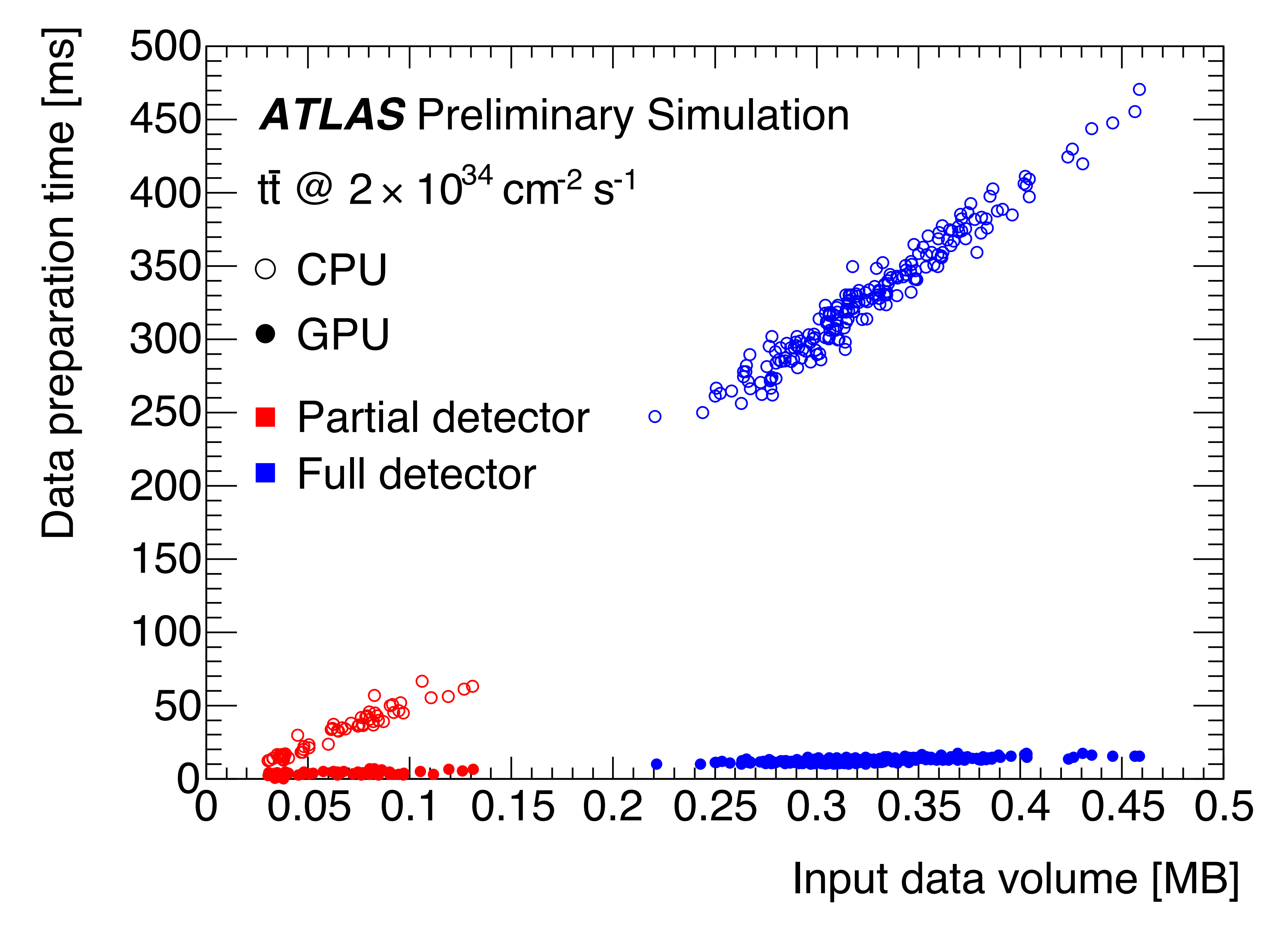
Parallel Clone Removal

- Difficult due to number of track pairs: $N \times (N - 1) / 2$
- Separate into two steps:
  - Identification/merging of clones - same extension spacepoints, different seeds



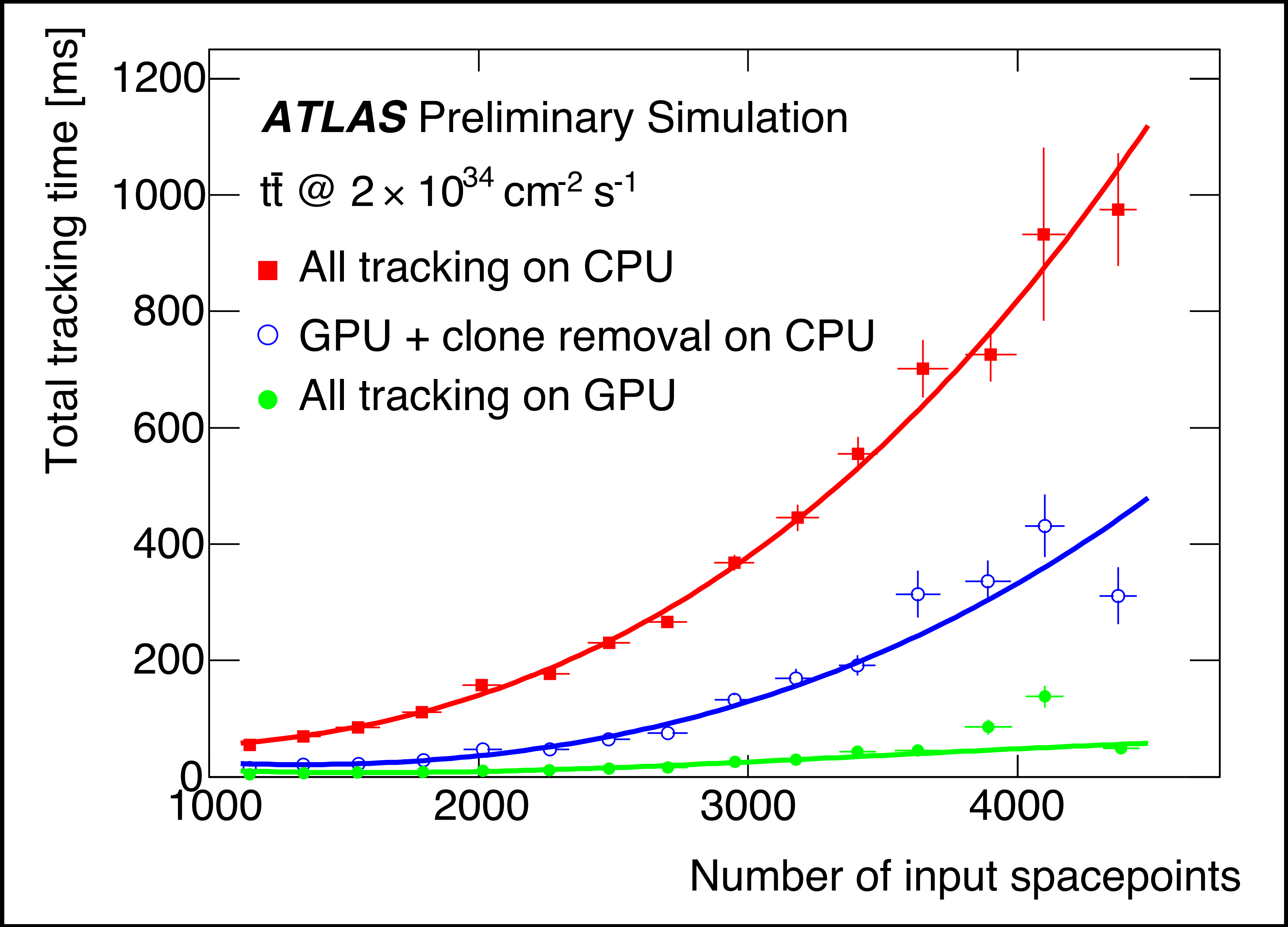
  - Removal of fake tracks
- Each GPU thread handles a range of tracks
- Stored in global memory, slow, but gain due to high number of track candidates

Data Preparation Results



Bytestream decoding and clustering show a **26x** speed-up on NVIDIA C2050 GPU vs single-threaded Intel E5620 CPU

Tracking Results



Track formation and clone removal show a **12x** speed-up on NVIDIA C2050 GPU vs single-threaded Intel E5620 CPU

CUDA Device Performance Comparison

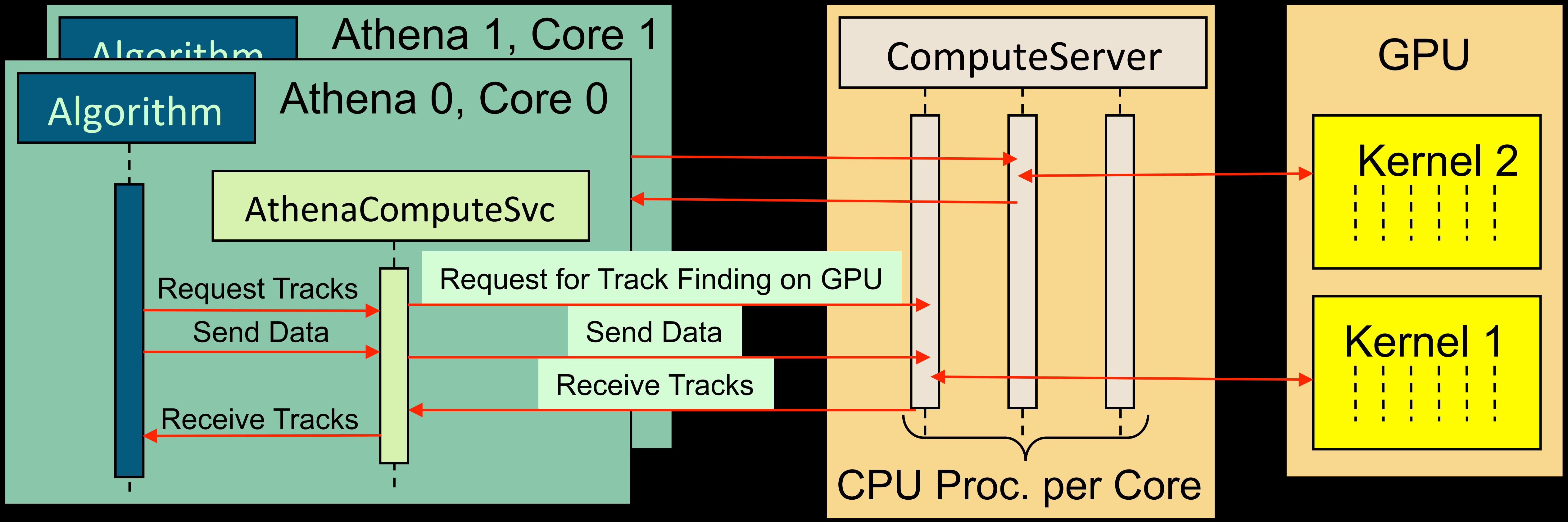
Device	Architecture	Cores	Core Speed	Processing Time	Processing Rate
C1060	Tesla	240	1300 MHz	26.8 ms	37.3 Hz
GT 630M	Fermi	96	800 MHz	18.3 ms	54.5 Hz
GT 650M	Kepler	384	835 MHz	17.2 ms	58.2 Hz
C2050	Fermi	448	1150 MHz	9.87 ms	101 Hz
K20	Kepler	2496	706 MHz	7.83 ms	128 Hz
K40	Kepler	2880	745 MHz	6.39 ms	156 Hz

Porting Complexities

Problem	Solution
Existing implementations serial	2 years development, 50% time
Complex spaghetti code structure	
Minimal documentation/comments	
Complex C++ inheritance structure	
Large, complex hardware maps	Special global hash tables developed for in-memory lookup
Custom build system for trigger, difficult to integrate CUDA	Developed client-server architecture for shared GPU operation
Multiple trigger instances usually run on each node	

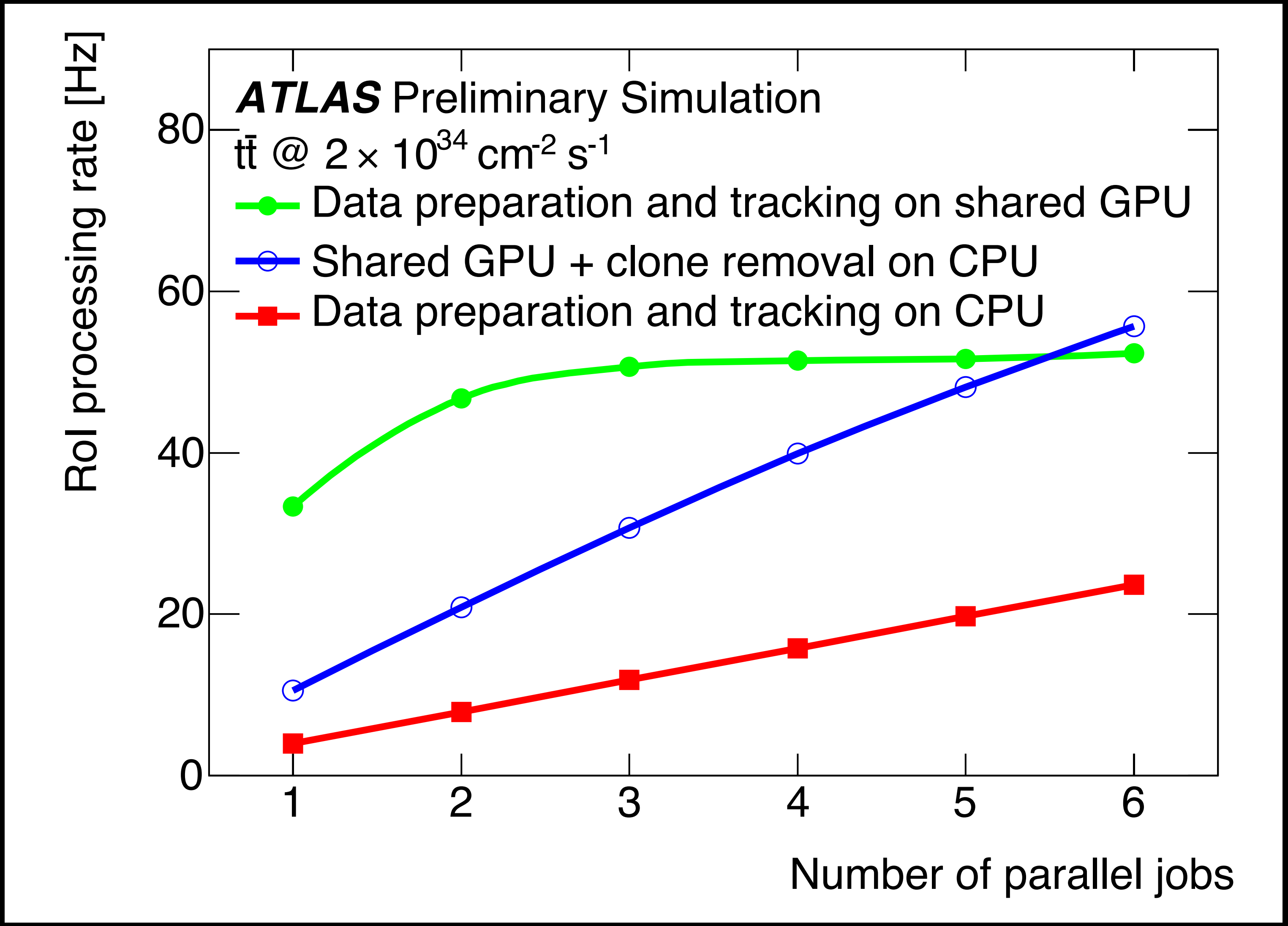
Client-Server Architecture

- Client-server architecture allows GPU resources to be shared amongst multiple trigger instances
- Data transfer is done over shared memory segment
  - Also used as CUDA host buffer
- Minimizes integration surface in trigger software - only POSIX required
- Allows for GPU memory resources (e.g. hardware maps) to be shared



Client-Server Performance

- Client-server architecture appears feasible
- Performance does seem to saturate with number of trigger jobs
  - Likely due to a limited number of streaming multiprocessors
- In practice, other requirements of trigger software impose more immediate limitations on trigger instance count



OpenCL Studies

- The CUDA implementation has been ported to OpenCL
- Initial performance comparisons show encouraging results on GPU, ~15% performance loss on the C2050
- Disparate results on heterogeneous hardware

Operation	NVIDIA C2050 (CUDA)	NVIDIA C2050 (OpenCL)	AMD FirePro GPU (OpenCL)	Dual 16-Core AMD 6276 CPU (OpenCL)
Pixel Processing	3.2 ms	3.9 ms	8.3 ms	19.0 ms
SCT Processing	3.6 ms	4.0 ms	7.7 ms	12.1 ms
Total Processing	6.8 ms	7.9 ms	16 ms	31.1 ms

Conclusion and Outlook

- GPUs show enormous promise for optimization of ATLAS trigger algorithms
- No free lunch - GPU porting is non-trivial and suitable replacement algorithms non-obvious
- GPU programming requires a significant amount of per-device optimization for maximal performance
- Code will be expanded to include muon and calorimeter data, as well as jet reconstruction
- Main obstacles to deployment: cooling, code-base integration and portability, heterogeneous hardware