Towards Chip-on-Chip Neuroscience

Fast Mining of Neuronal Spike Streams Using Graphics Hardware

Yong Cao,
Debprakash Patnaik, Sean Ponce, Jeremy Archuleta,
Patrick Butler, Wu-chun Feng, and Naren Ramakrishnan

Virginia Polytechnic Institute and State University

VirginiaTech
Invent the Future

Motivation

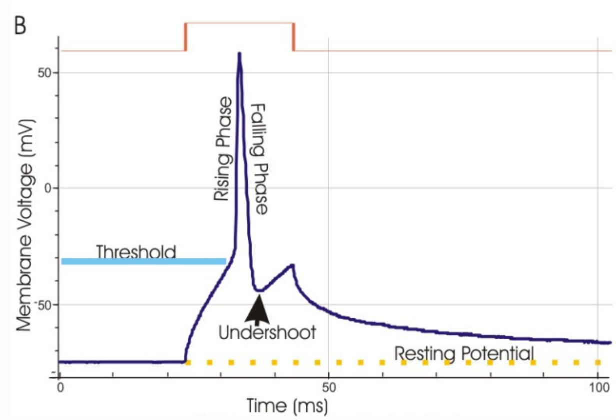
- **Reverse-engineer the brain** GRAND CHALLENGES FOR ENGINEERING

National Academy of Engineering Top 5 Grand Challenges

Cited from Sciseek.com

Question:

How are the neurons connected?



Action Potentials (Spikes)

Motivation

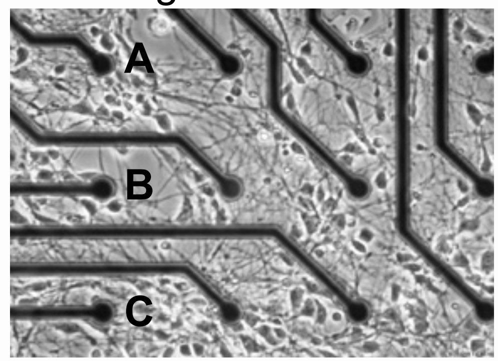
◆Reverse-engineer the brain

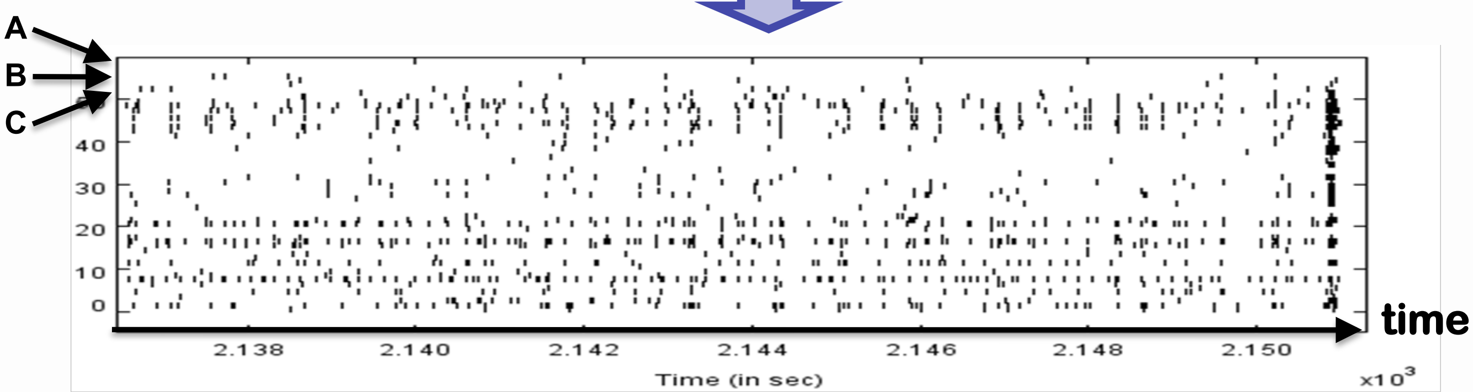
GRAND CHALLENGES FOR ENGINEERING

National Academy of Engineering Top 5 Grand Challenges

Multi-Electrode Array (MEA)

Neurons grown on MEA Chip





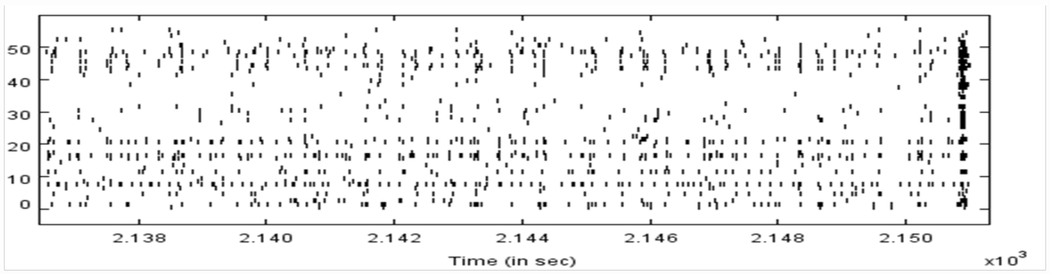
Spike Train Stream

Motivation

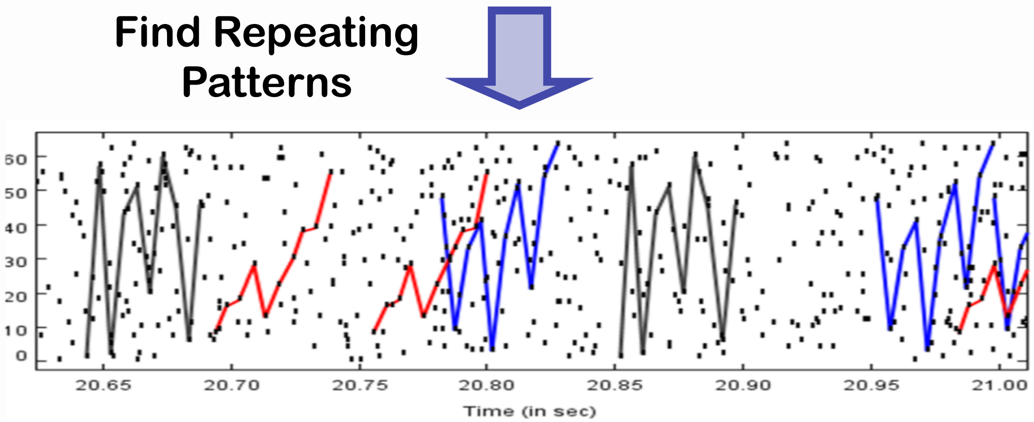
◆ **Reverse-engineer the brain**

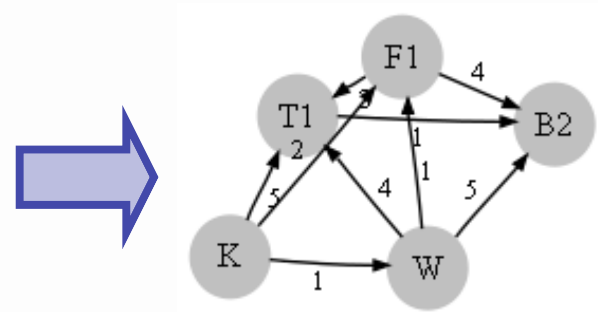
GRAND CHALLENGES FOR ENGINEERING

National Academy of Engineering Top 5 Grand Challenges



Find Repeating Patterns



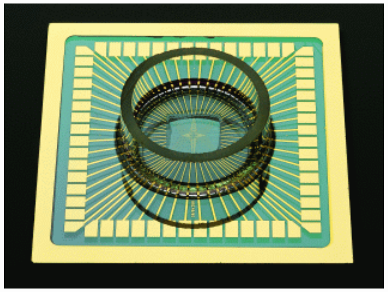


Infer Network Connectivity

Contributions

- Fast data mining of spike train stream on Graphics Processing Units (GPUs)

MEA Chip



Multi-Electrode Array (MEA)

GPU Chip

NVIDIA GTX280 Graphics Card

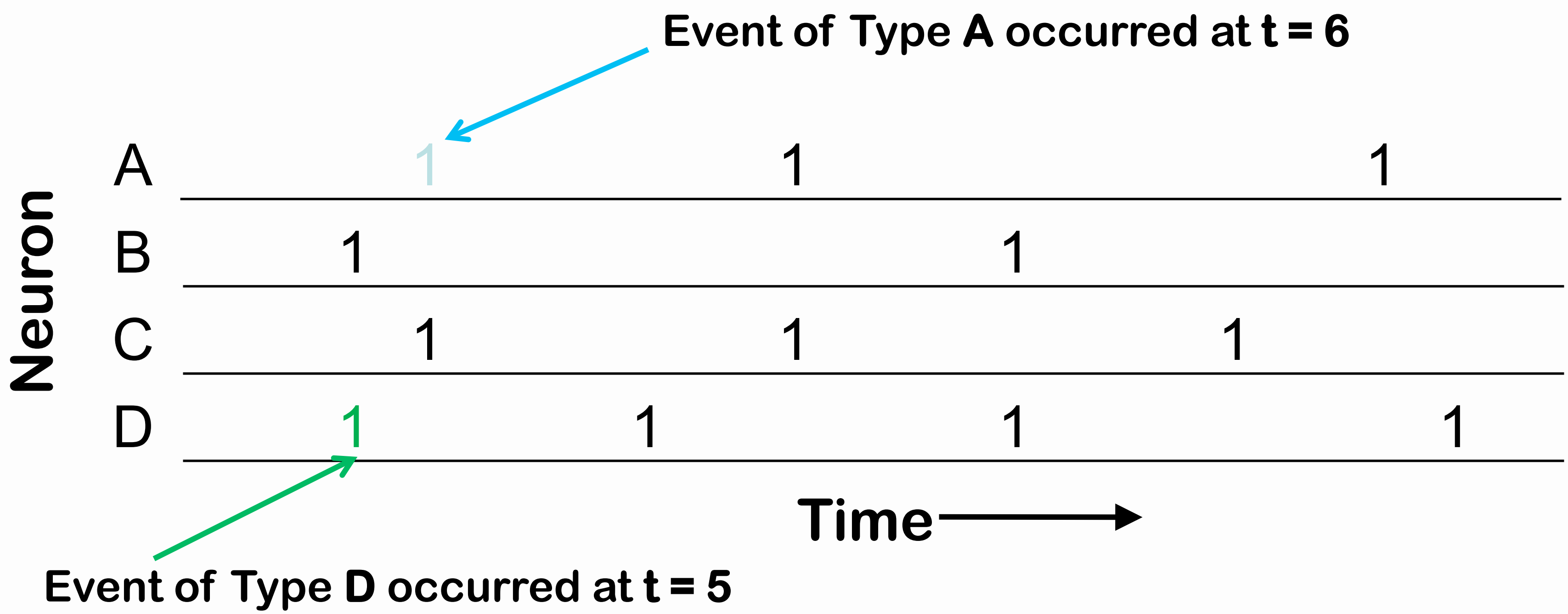
Contributions

- Fast data mining of spike train stream on Graphics Processing Units (GPUs)
- Two key algorithmic strategies to address scalability problem on GPU
  - A hybrid mining approach
  - A two-pass elimination approach

Background

- Event stream data: sequence of neurons firing

$$\langle (E_1,t_1),(E_2,t_2),\ldots,(E_n,t_n)\rangle$$



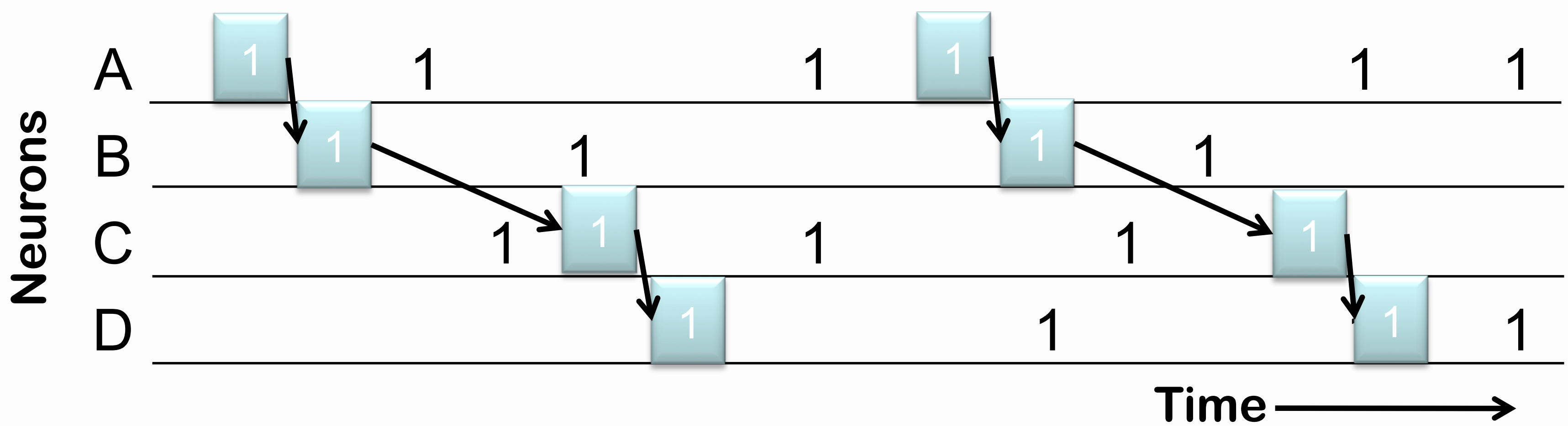
Background

- **Pattern or Episode**

Inter-event constraint

$$A \xrightarrow{(0,5]} B \xrightarrow{(5,10]} C \xrightarrow{(0,5]} D$$

- **Occurrences (Non-overlapped)**



Episode appears twice in the event stream.

Background

- **Data mining problem:**

  Find all possible episodes / patterns which occur more than X-times in the event sequence.

- **Challenge:**
  - **Combinatorial Explosion: large number of episodes to count**

Episode Size/Length:	1	2	3	4	......
	A	$A \rightarrow B$	$A \rightarrow B \rightarrow C$	$A \rightarrow B \rightarrow C \rightarrow D$	
	B	$B \rightarrow A$	$A \rightarrow C \rightarrow B$	$A \rightarrow C \rightarrow B \rightarrow D$	
	$\vdots$	$A \rightarrow C$	$B \rightarrow A \rightarrow C$	$A \rightarrow C \rightarrow D \rightarrow B$	
		$\vdots$	$B \rightarrow C \rightarrow A$	$A \rightarrow D \rightarrow B \rightarrow C$	
			$\vdots$	$A \rightarrow D \rightarrow C \rightarrow B$	
				$\vdots$	

Background

◆Mining Algorithm

(A level wise procedure to control combinatorial explosion)

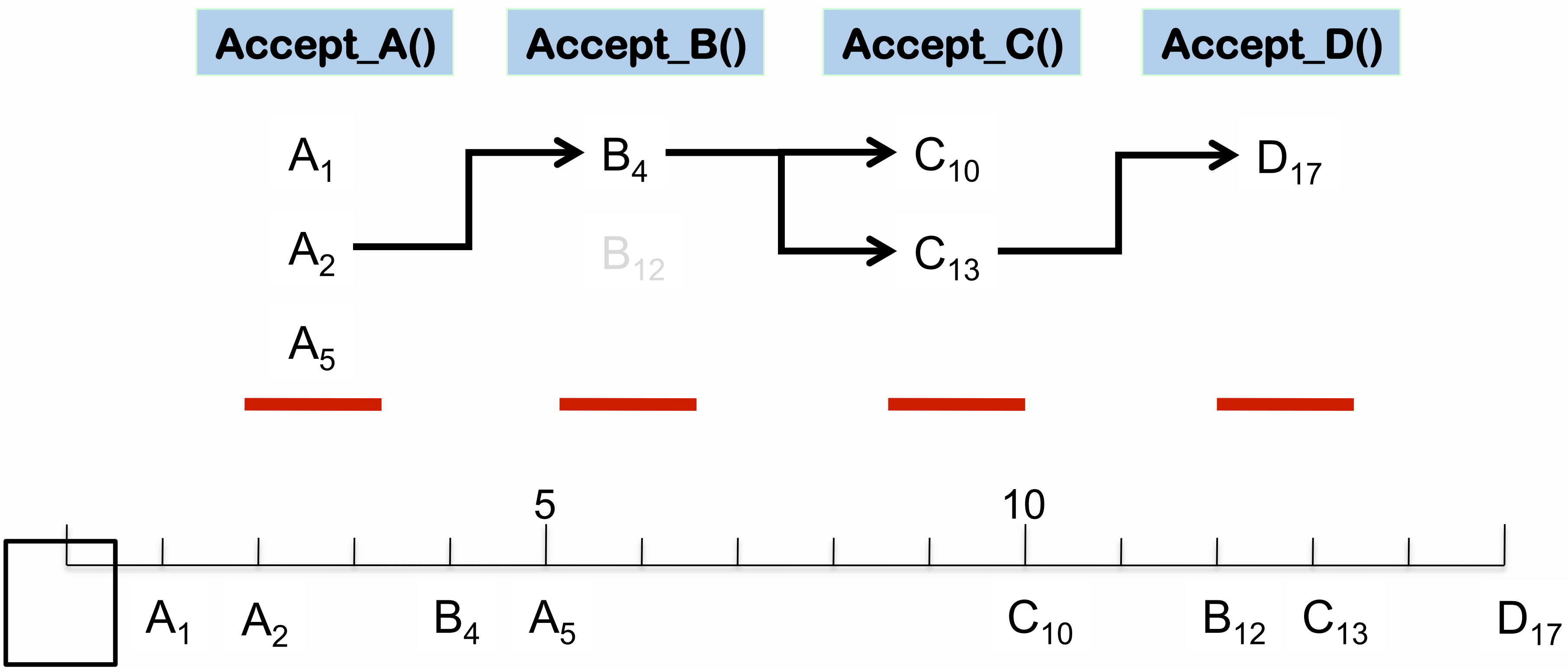
- Generate an initial list of candidate *size-1* episodes
- Repeat until - no more candidate episodes
  - **Count**: Occurrences of *size-M* candidate episodes
  - Prune: Retain only frequent episodes
  - Candidate Generation: *size-(M+1)* candidate episodes from N-size frequent episodes
- Output all the frequent episodes

Computational bottleneck

Background

- Counting Algorithm (for one episode)

Episode: $A \xrightarrow{(0,5]} B \xrightarrow{(5,10]} C \xrightarrow{(0,5]} D$



Event Stream

Problem Statement

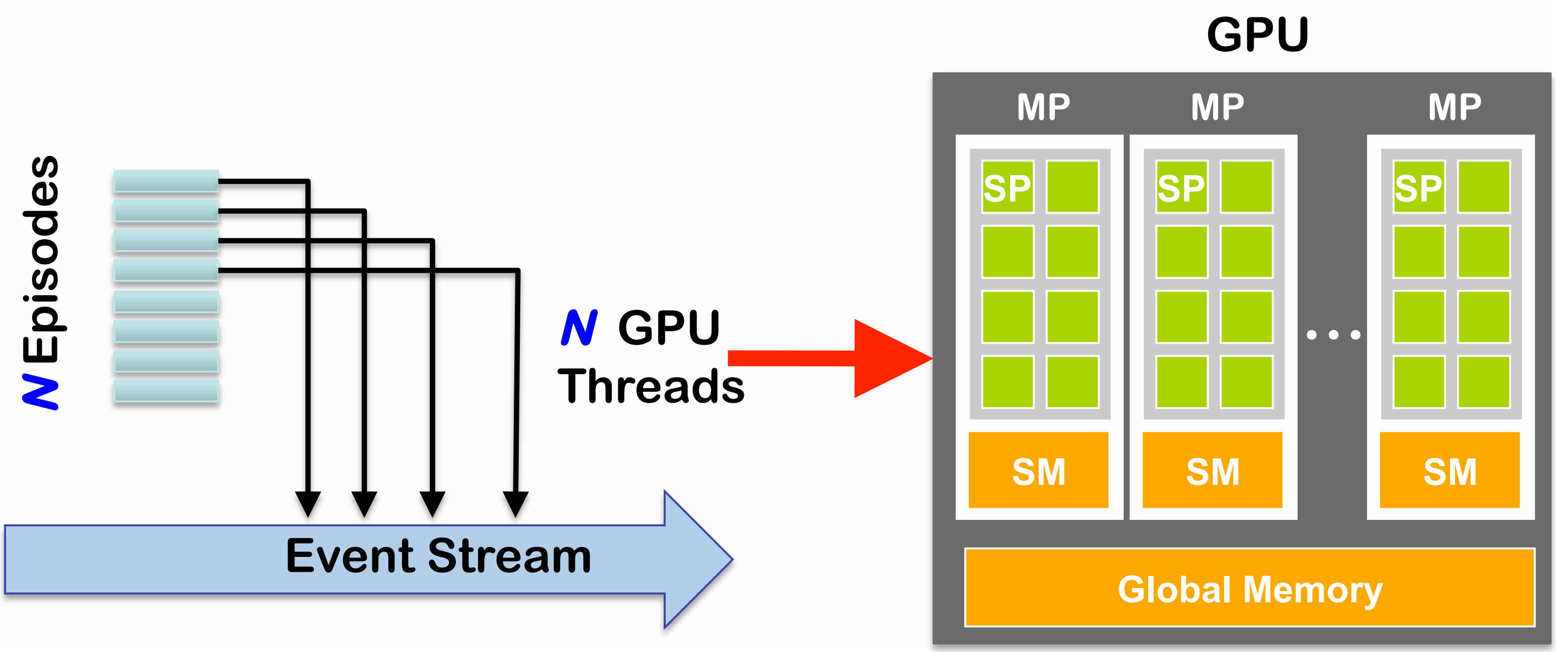
- Find an efficient counting algorithm on GPU to count the occurrences of *N* *size-M* episodes in an event stream.

- Address scalability problem on GPU's massive parallel execution architecture.

VirginiaTech Invent the Future

A Naïve Approach

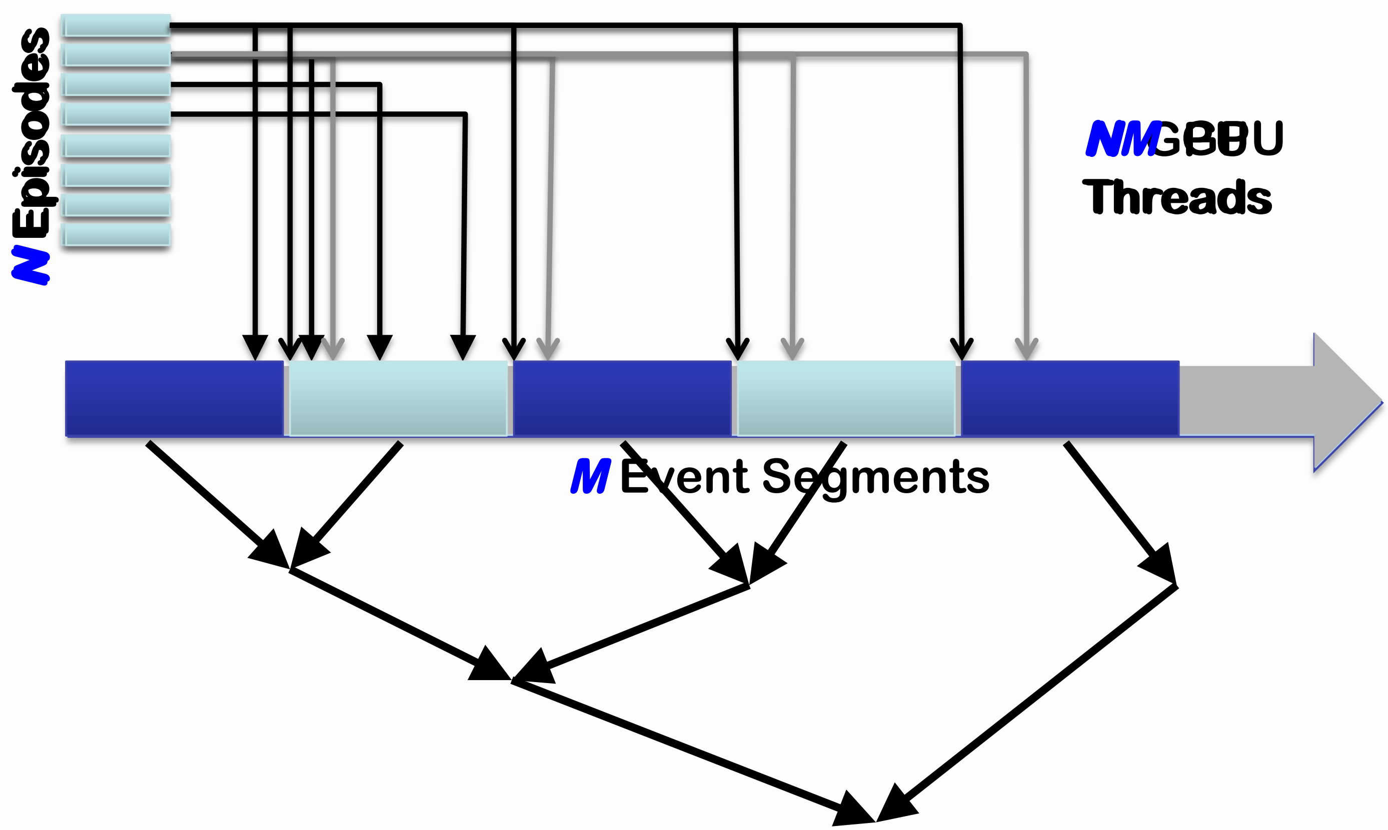
- One episode per GPU thread (PTPE)
  - Each thread counts one episode
  - Simple extension of serial counting



- **Efficient when the number of episode is larger than the number of GPU cores.**

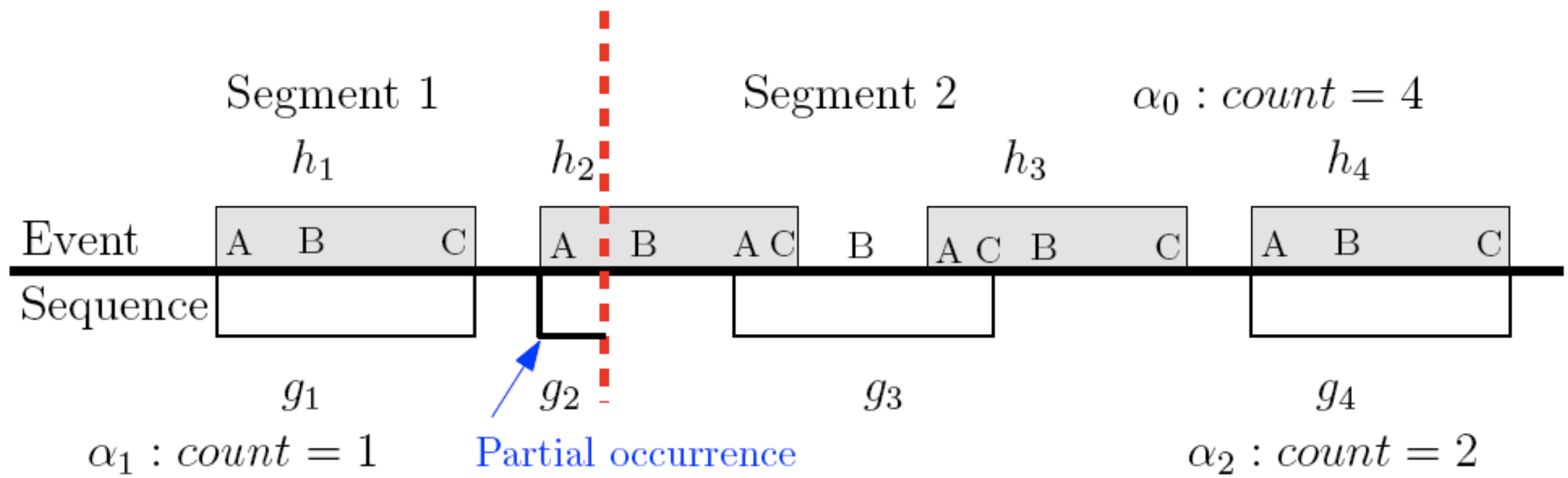
Small Scale

- Not enough episodes/thread, some GPU cores will be idle.
- Solution: Increase the level of parallelism.
  Multiple Thread per Episode (MTPE)



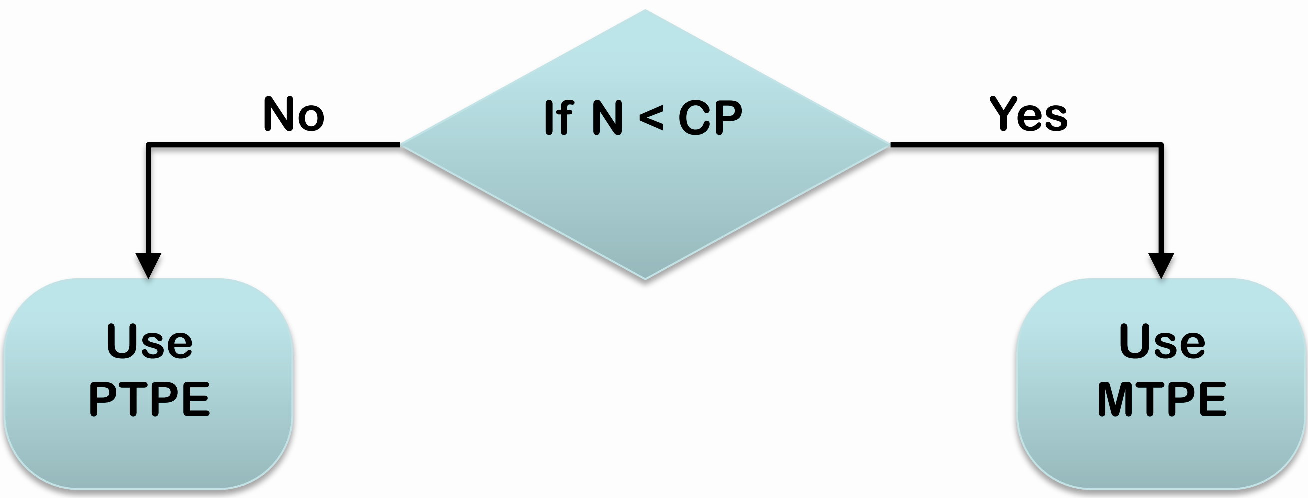
Small Scale

- Problem with simple count merge.



A Hybrid Approach

- Choose the right algorithm with respect to the number of episodes ***N***.
- Define a switching threshold - **Crossover point** (CP)



GPU computing capacity

$$CP = \boxed{MP \times B_{MP} \times T_B} \times \boxed{f(size)}$$

Performance Penalty Factor

MP : Number of multi-processors

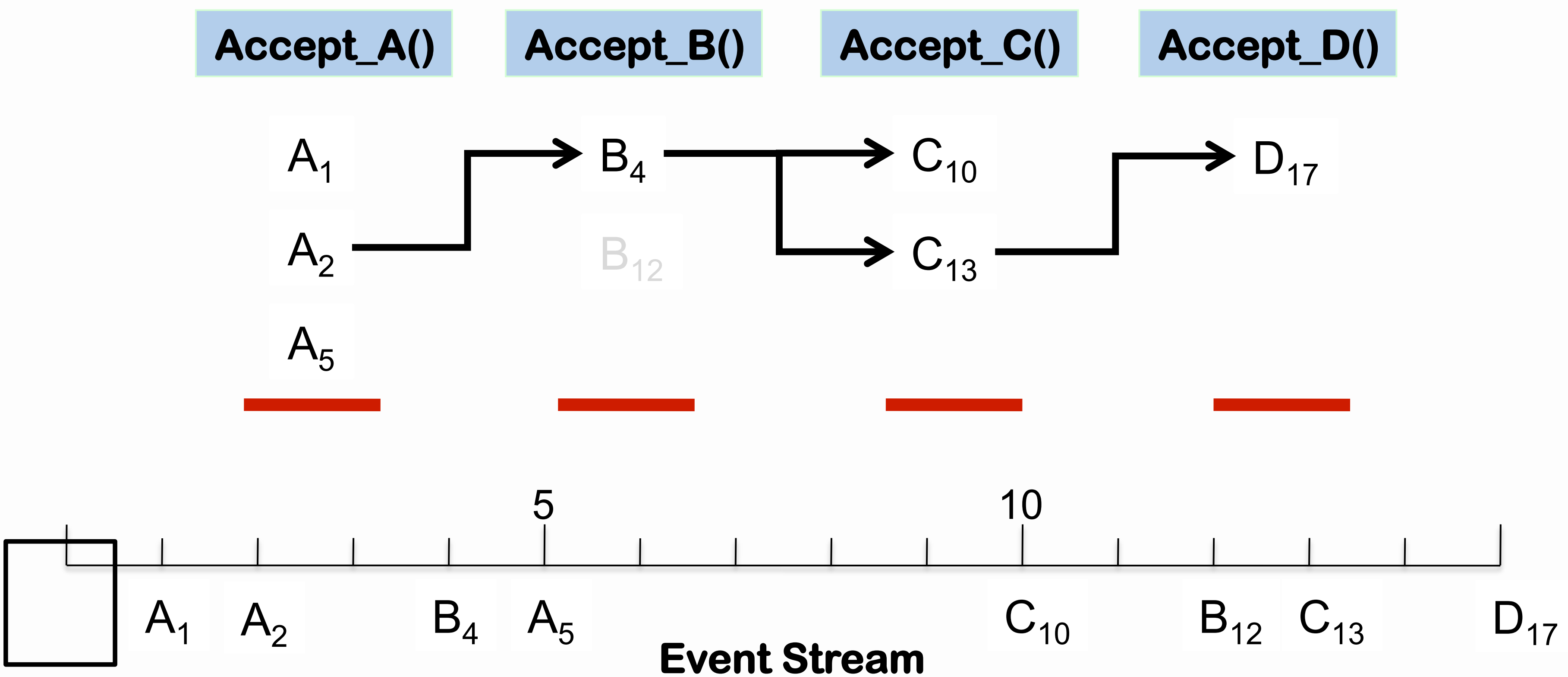
B_{MP} : Block per multi-processor

T_B : Thread per block

Large Scale

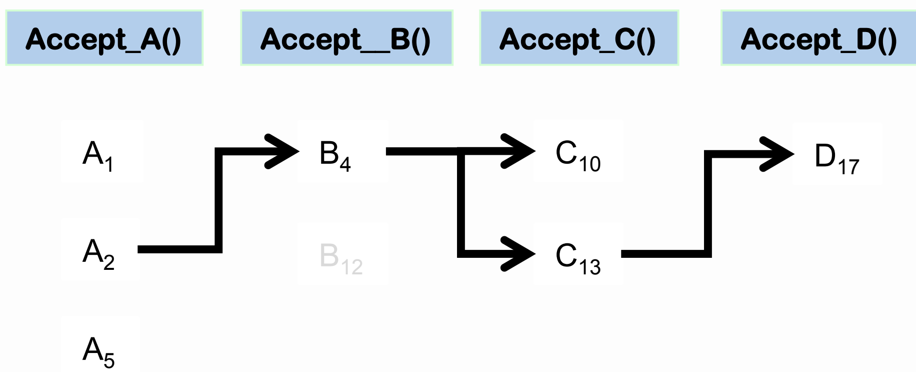
- Problem: Original counting algorithm is too complex for a GPU kernel function.

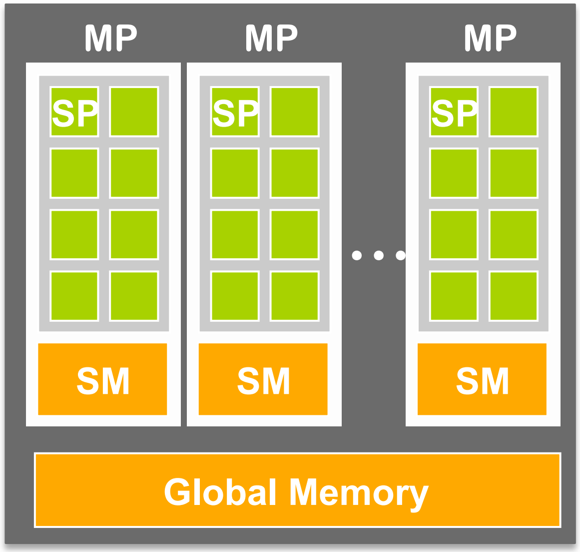
Episode: $A \xrightarrow{(0,5]} B \xrightarrow{(5,10]} C \xrightarrow{(0,5]} D$



Large Scale

- Problem: Original counting algorithm is too complex for a GPU kernel function.





- Large shared memory usage
- Large register file usage
- Large number of branching instructions

Large Scale

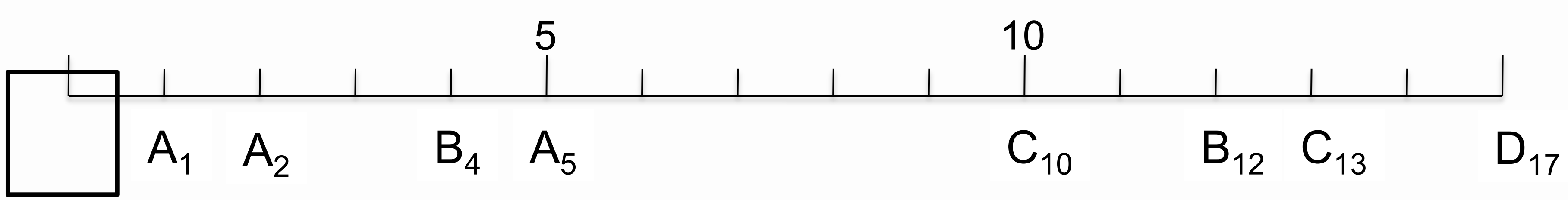
- Solution: *PreElim* algorithm
  - Less constrained counting → Simple kernel function
  - Upper bound only

Episode: $A \xrightarrow{(-,5]} B \xrightarrow{(-,10]} C \xrightarrow{(-,5]} D$

Accept_A()	Accept_B()	Accept_C()	Accept_D()

$A_5 \longrightarrow B_4 \longrightarrow C_{13} \longrightarrow D_{17}$

B_{12}



Event Stream

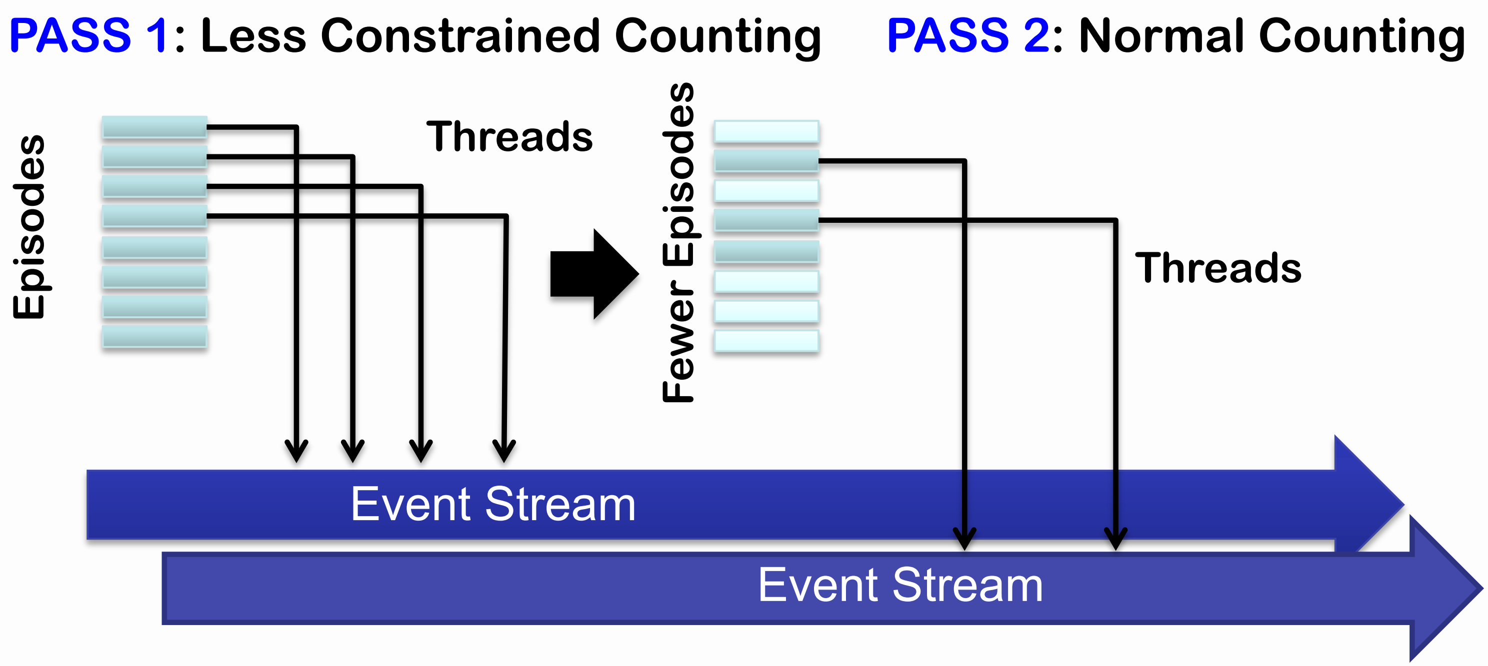
Large Scale

- A simpler kernel function

	Shared Memory	Register	Local Memory
PreElim	4 x Episode Size	13	0
Normal Counting	44 x Episode Size	17	80

Large Scale

- **Solution:**
  - Two-pass elimination approach



Large Scale

- A simpler kernel function

Compile Time Difference

	Shared Memory	Register	Local Memory
PreElim	4 x Episode Size	13	0
Normal Counting	44 x Episode Size	17	80

Run Time Difference

	Local Memory Load and Store	Divergent Branching
Two Pass	24,770,310	12,258,590
Hybrid	210,773,785	14,161,399

Results

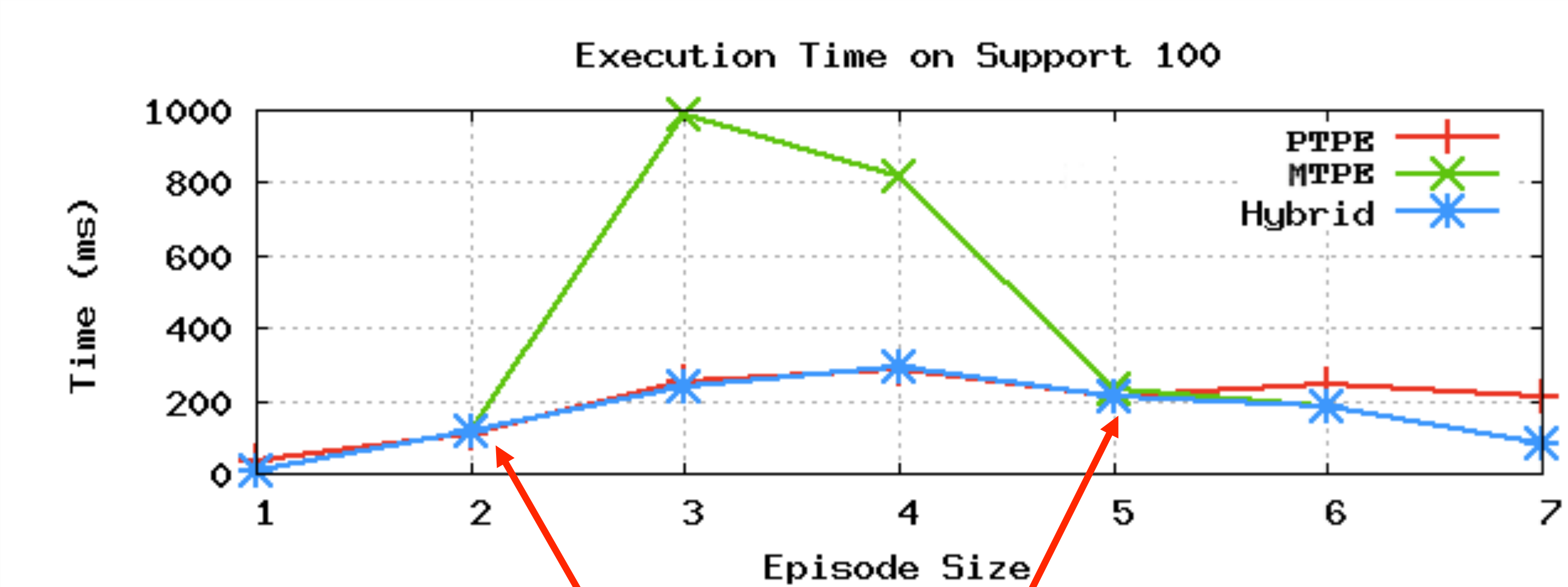
- Hardware
  - Computer (custom-built)
    - Intel Core2 Quad @ 2.33GHz
    - 4GB memory
  - Graphics Card (Nvidia GTX 280 GPU)
    - 240 cores (30 MPs * 8 cores) @ 1.3GHz
    - 1GB global memory
    - 16K shared memory for each MP

Results

- Datasets
  - Synthetic (*Sym26*)
    - 60 seconds with 50,000 events
  - Real (Culture growing for 5 weeks)
    - Day 33: *2-1-33* (333478 events)
    - Day 34: *2-1-34* (406795 events)
    - Day 35: *2-1-35* (526380 events)

Results:

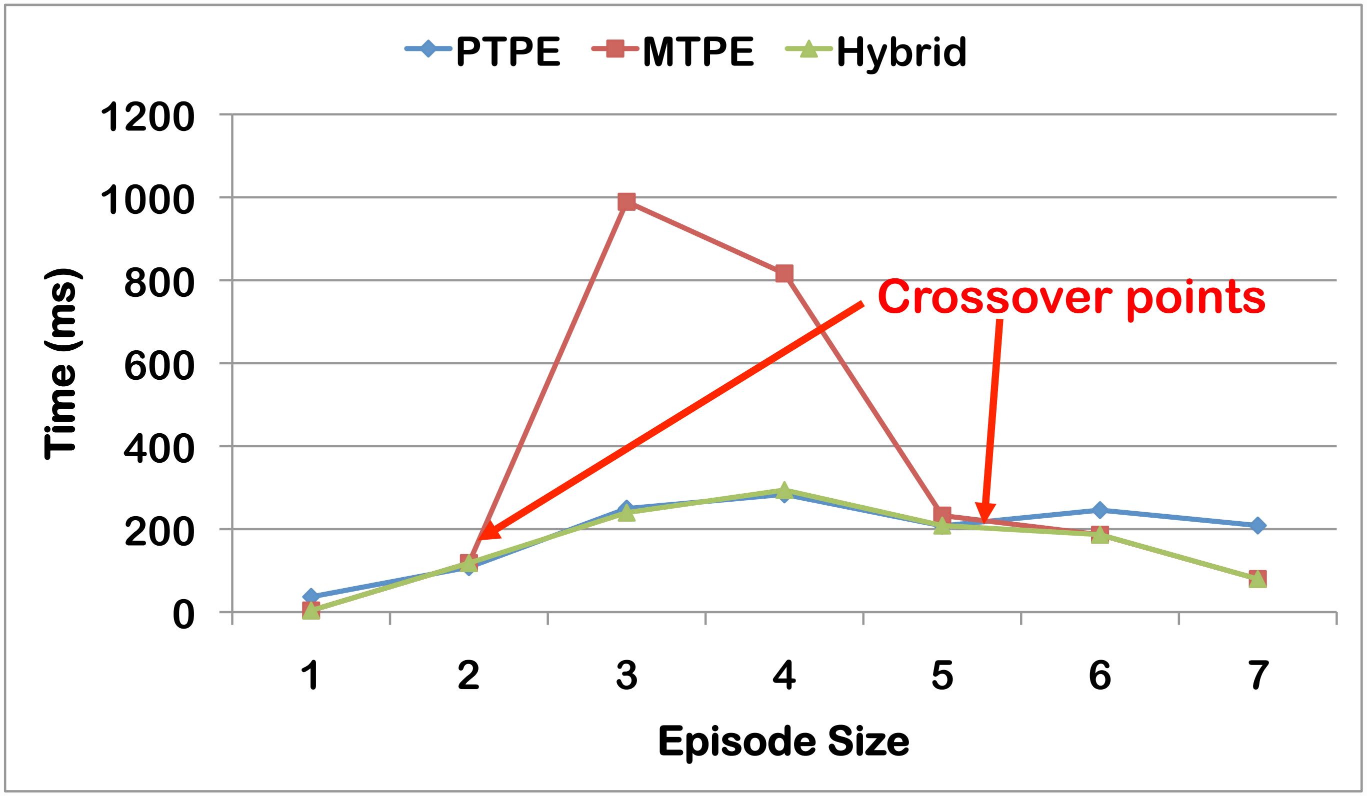
◆ **PTPE vs MTPE**



Crossover points

Results:

- Performance of the Hybrid Approach

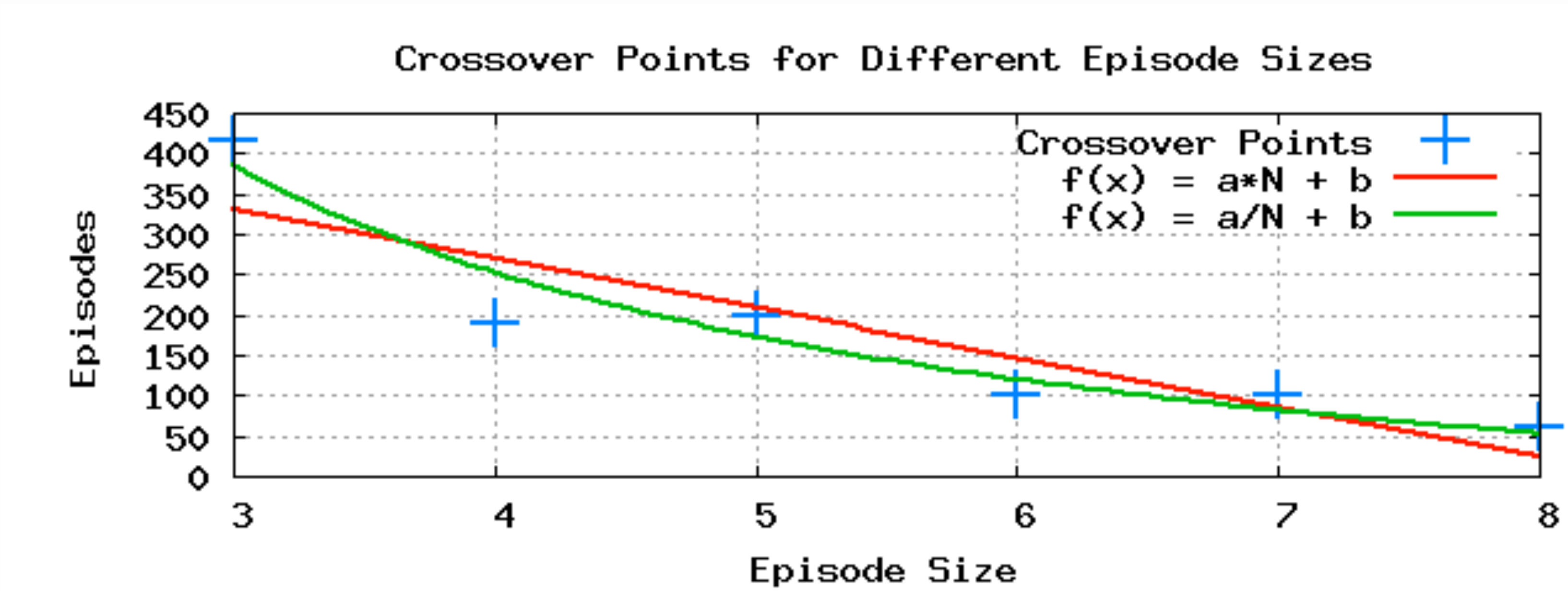


Episode Number:	26	650	4075	1288	228	63	3

Sym26 dataset, Support = 100

Results:

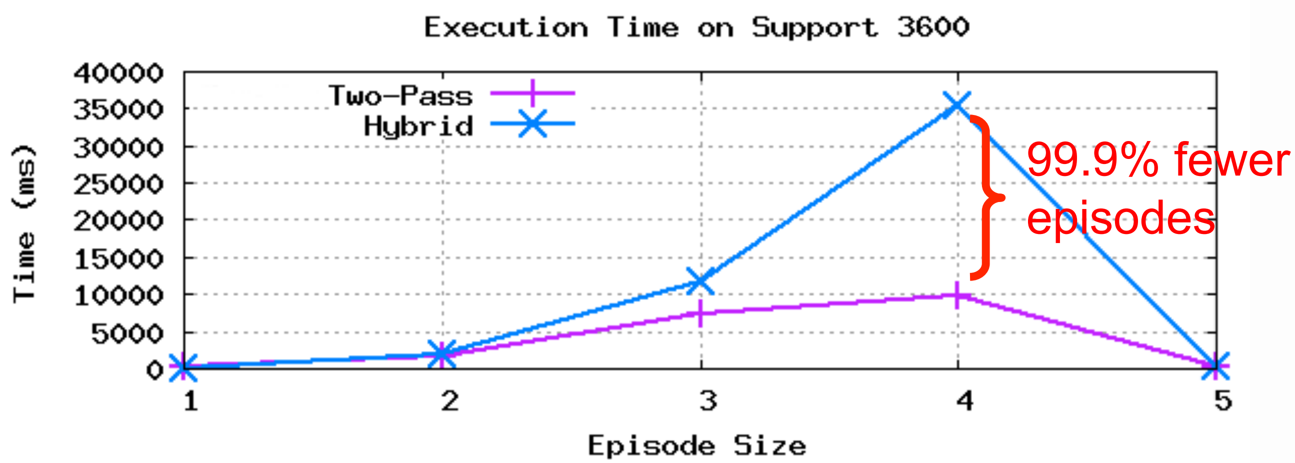
- Crossover Point Estimation

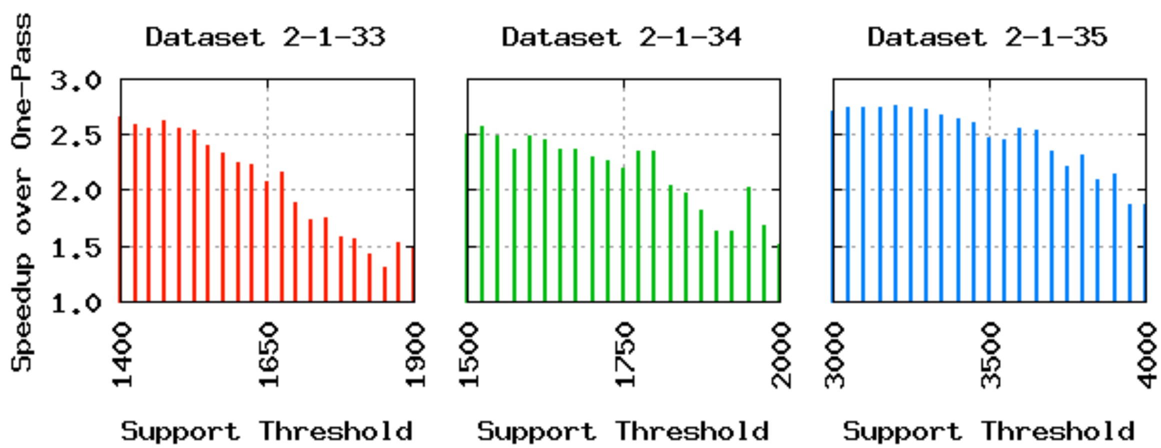


- $f(size) = \frac{a}{size} + b$ is a better fit.
- A least square fit is performed.

Results:

◆Two-pass approach vs Hybrid approach

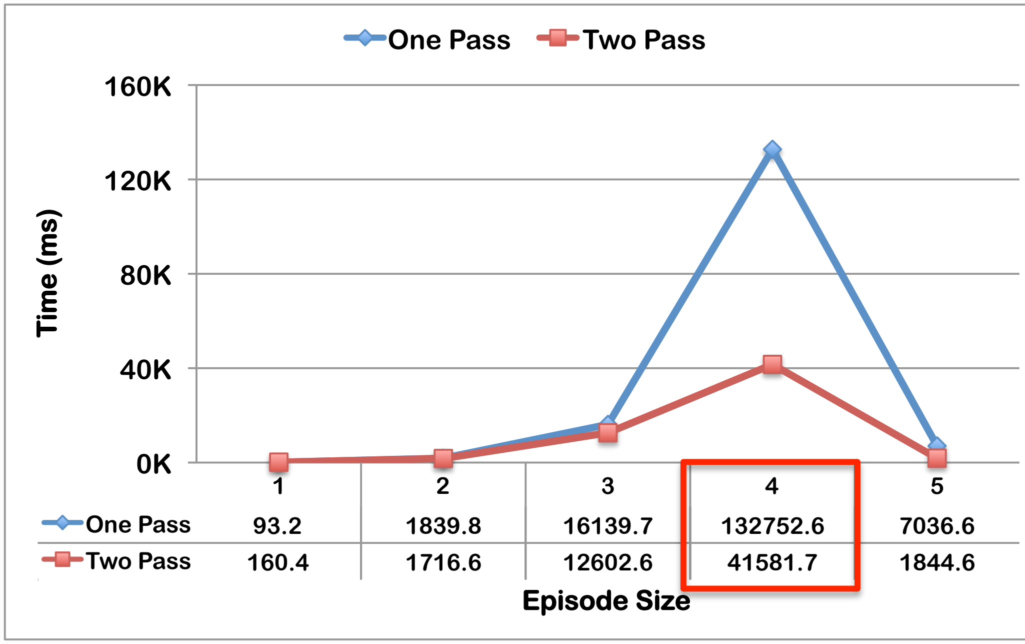




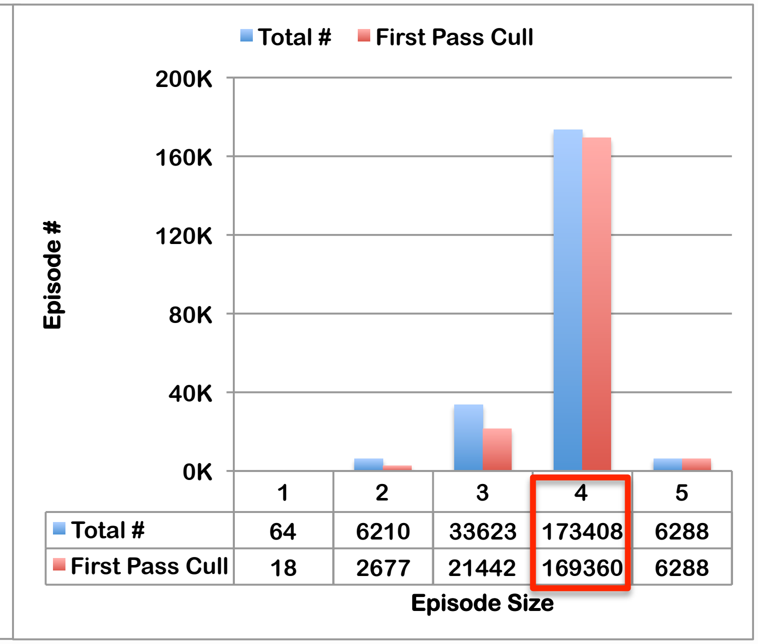
VirginiaTech
Invent the Future

Results:

- Performance of the Two-pass approach



	1	2	3	4	5
One Pass	93.2	1839.8	16139.7	132752.6	7036.6
Two Pass	160.4	1716.6	12602.6	41581.7	1844.6



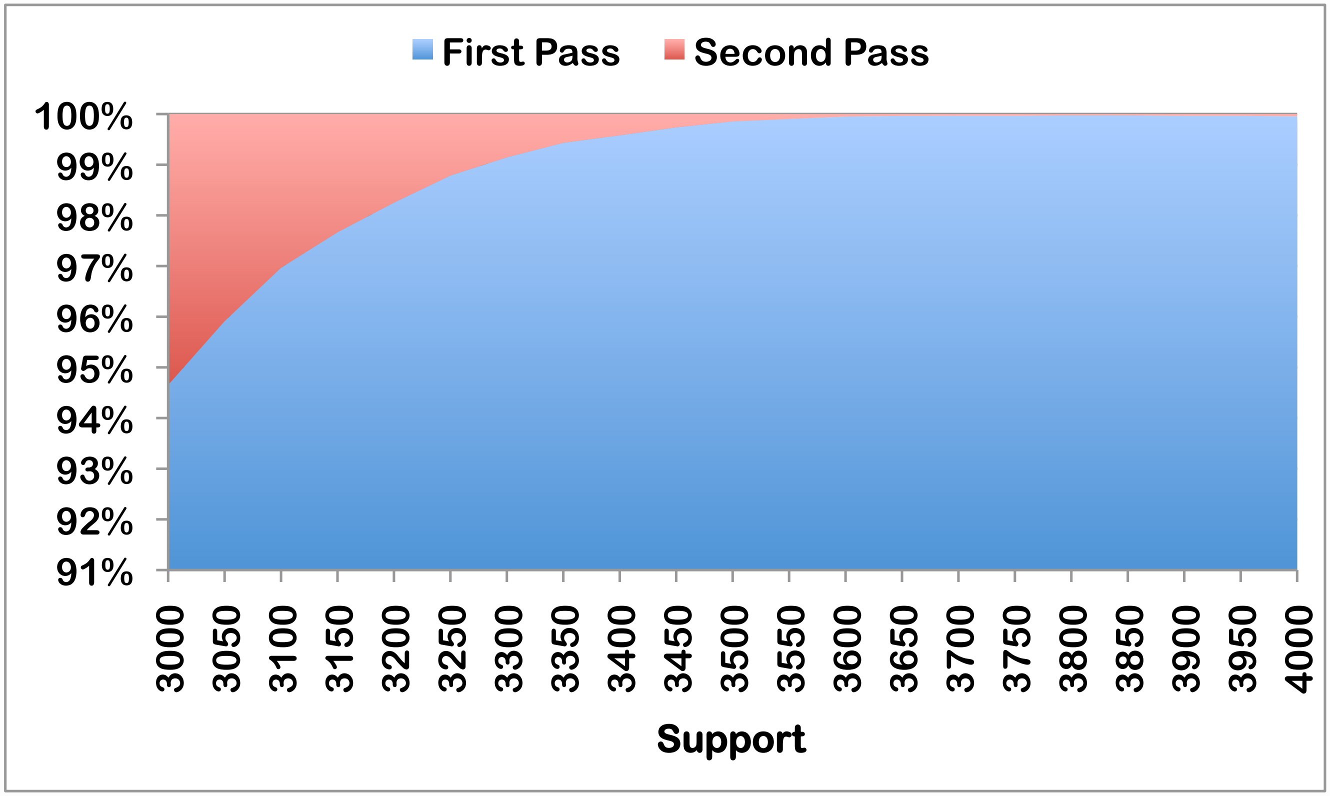
	1	2	3	4	5
Total #	64	6210	33623	173408	6288
First Pass Cull	18	2677	21442	169360	6288

2-1-35 dataset, Support = 3150

VirginiaTech
Invent the Future

Results:

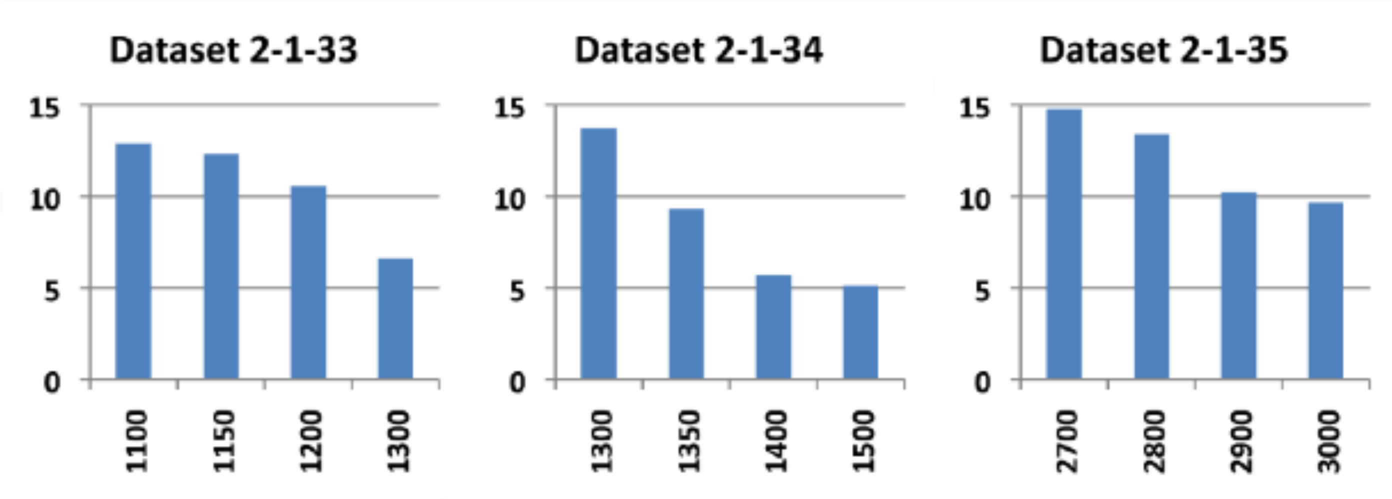
- Percentage of episodes eliminated by each pass



2-1-35 **dataset, episode size = 4**

Results:

◆GPU vs CPU



- GPU is always faster than CPU
  - 5x - 15x speedup
  - Fair comparison
    - Two-pass algorithm used
    - Maximum threading for both

Conclusion and future work

- Massive parallelism is required for conquering near exponential search space
  - GPU's far more accessible than high performance clusters
- Frequent episode mining – Not data parallel
  - Redesigned algorithm
- Framework for real-time and interactive analysis of spike train experimental data

Conclusion

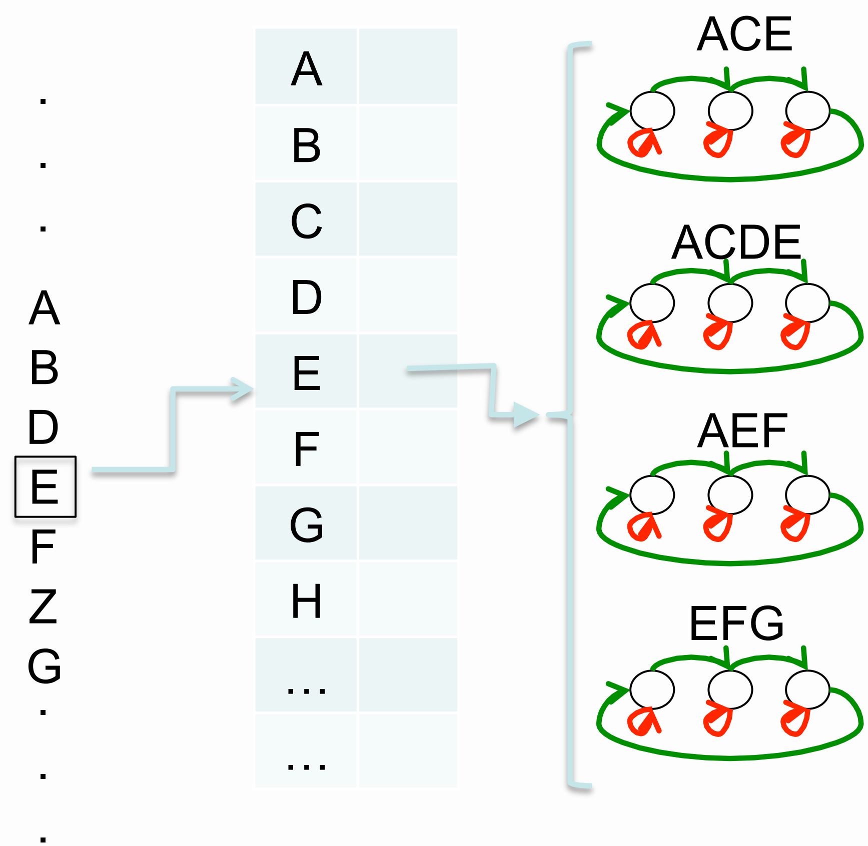
- A fast temporal data mining framework on GPUs
  - Commoditized system
  - Massive parallel execution architecture
- Two programming strategies
  - A hybrid approach
    - Increase level of parallelism (data segmentation + map-reduce)
  - Two-pass elimination approach
    - Decrease algorithm complexity (Task decomposition)

Thank you.

Questions.

CPU Implementation

- Parallel Execution via pthreads
- Optimized for CPU execution
  - Minimize disk access
  - Cache performance
- Implements Two-Pass Approach
  - PreElim – Simpler/ Quicker state machine
  - Full State Machine – Slower but is required to eliminate all unsupported episodes



Candidate Generation

- Level-wise
  - N-size frequent episodes => (N+1)-size candidates