WILLIAM & MARY
CHARTERED 1693

Architectural Support for Efficient Large-Scale Automata Processing

Hongyuan Liu* Mohamed Ibrahim* Onur Kayiran† Sreepathi Pai‡ Adwait Jog*
*College of William & Mary †AMD ‡University of Rochester

MICRO 2018

1- Automata Processing

Used widely in different areas

MACHINE LEARNING

Von Neumann architectures are not efficient at FSM processing

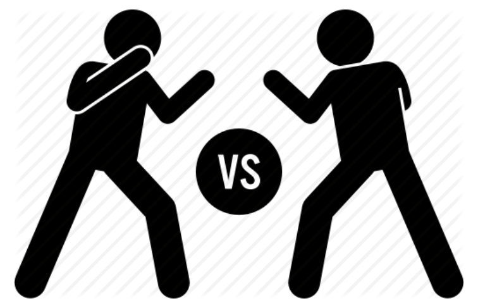
- ✖ Irregular memory accesses
- ✖ Limited Parallelism

Solution: Use Automata Processor (AP)

- ✓ Enables in-memory processing
- ✓ Exploits state parallelism of NFAs

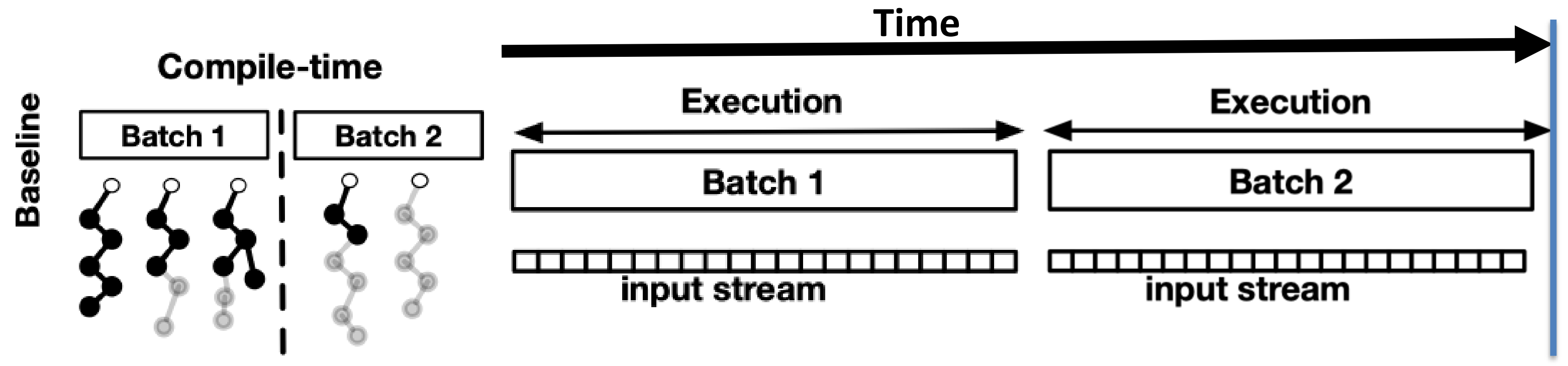
2- Challenges & Opportunities

Applications are getting Bigger



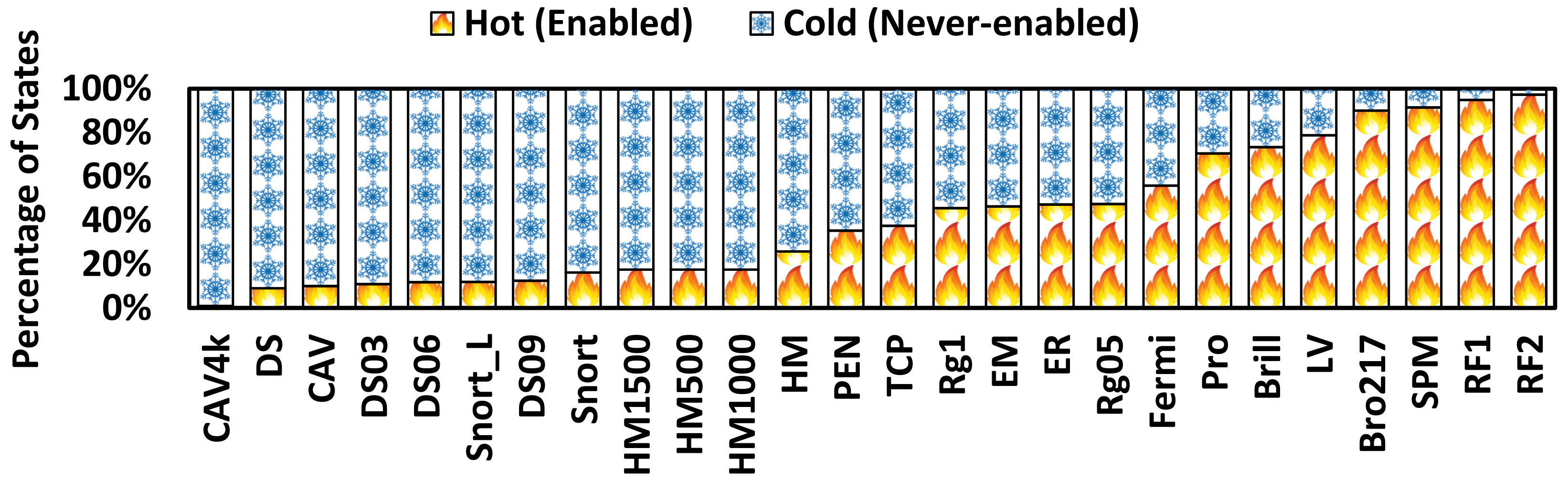
AP capacity is Limited

Challenge: Repeated Executions!



Opportunity: Underutilization of AP

Pattern mismatch → Many unused states are configured to AP

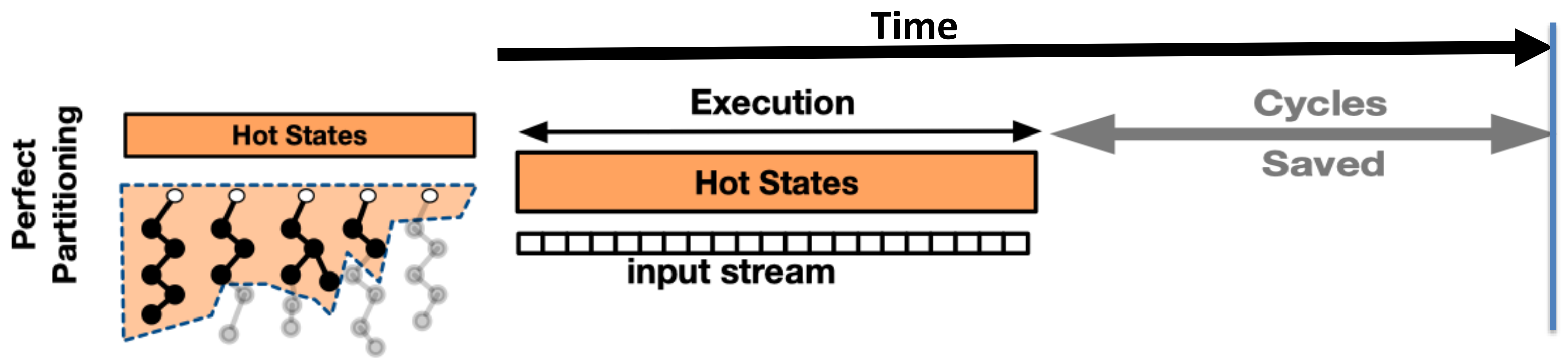


Potential Solution

Remove Cold states from the NFAs
Configure ONLY the Hot states to AP

Decrease Batches

3- Potential Benefits & Research Questions



✖ **Oracular knowledge of input**

✖ **Arbitrary states partitioning**

Question#1:
How to predict Cold states?

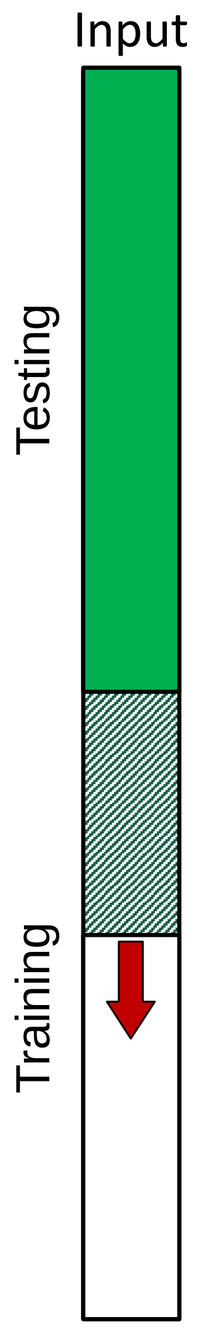
Question#2:
How to partition NFAs?

Mispredictions

Question#3:
How to handle mispredictions efficiently?

4- Efficient Automata Processing on AP

Q1: How to predict Cold states?



✖ **Oracular knowledge of input**

Solution: Use a small profiling input to predict the Hot/Cold states

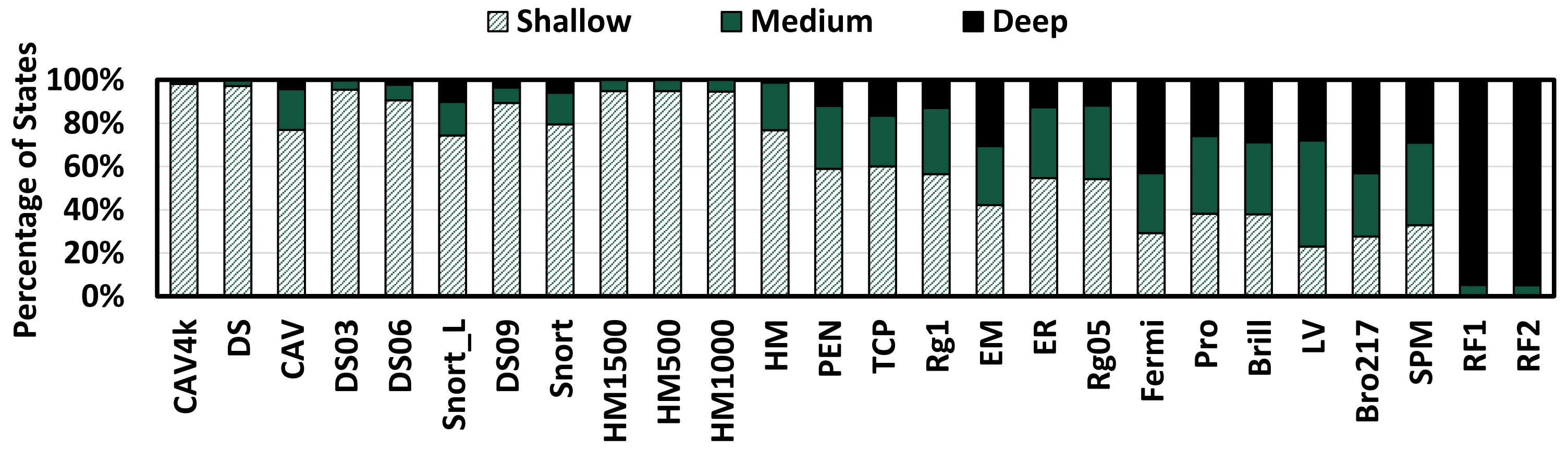
% from Input	% from Training	Accuracy
50%	100%	97%
10%	20%	93%
1%	2%	90%
0.1%	0.2%	87%

Q2: How to partition NFAs?

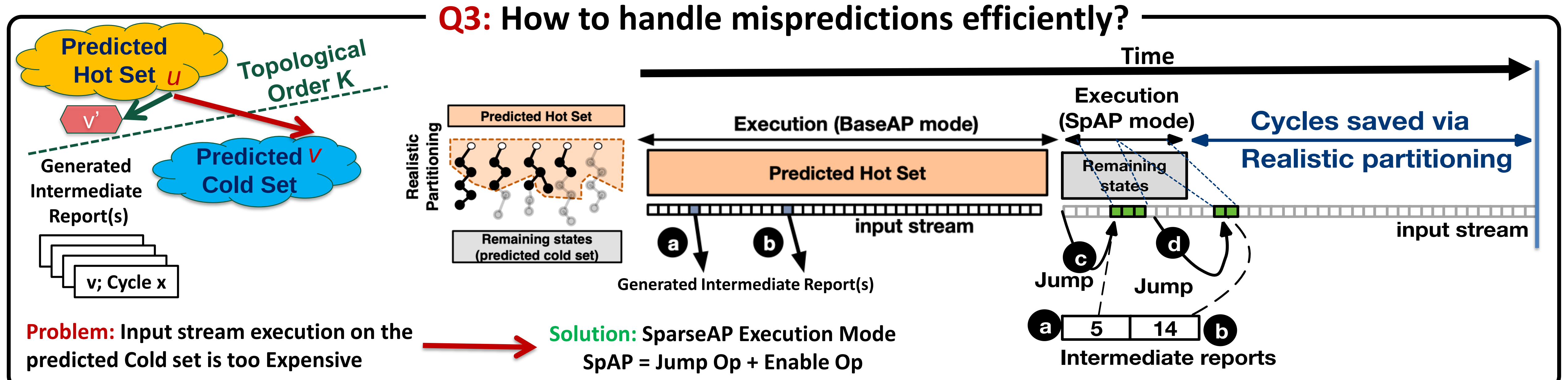
✖ **Arbitrary states partitioning**

Solution: Partition using Topological Order

- ✓ Correlates with Cold and Hot states
- ✓ Makes transition unidirectional

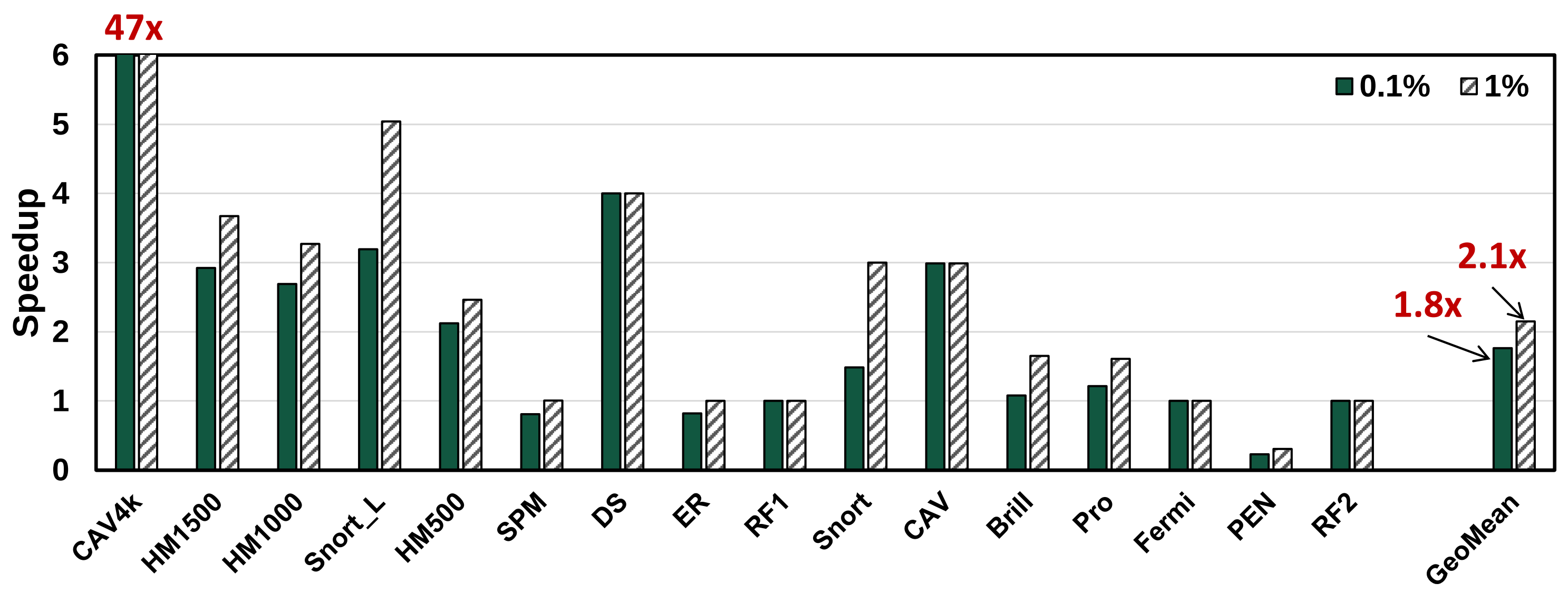


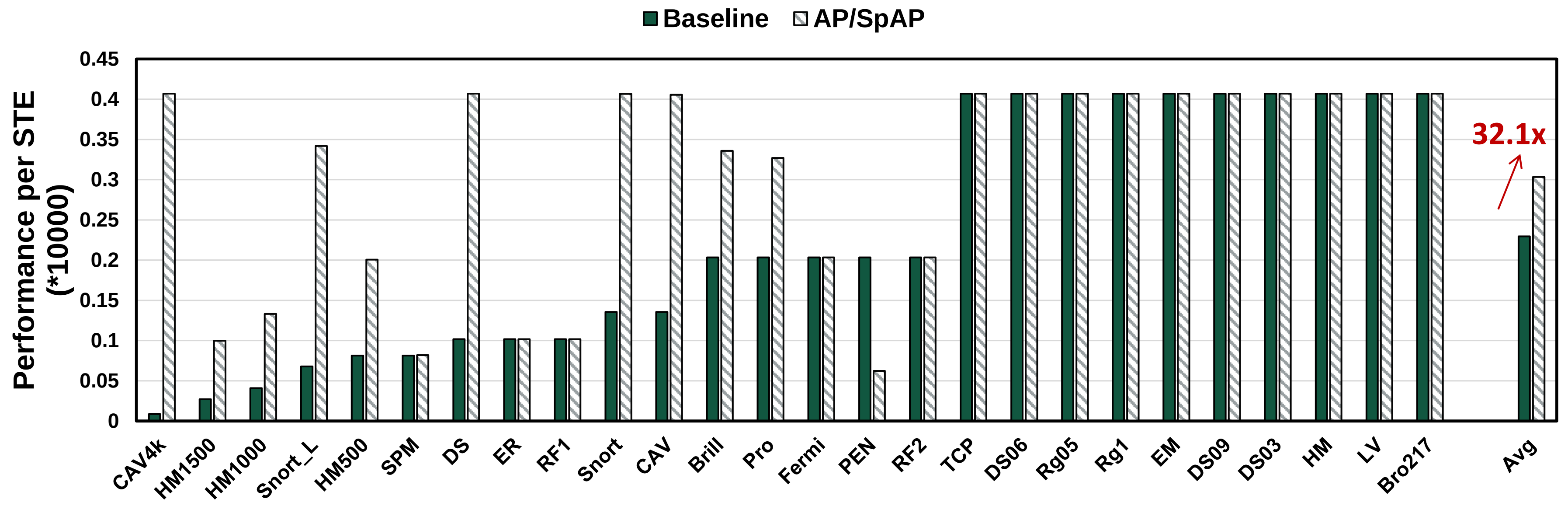
Q3: How to handle mispredictions efficiently?



Problem: Input stream execution on the predicted Cold set is too Expensive

Solution: SparseAP Execution Mode
SpAP = Jump Op + Enable Op





5- Summary

- **Observation:** Repeated configurations and executions on AP which causes inefficiency
- **Goal:** Accelerate large-scale NFA processing on AP
  - + Demonstrate that a large number of NFA states are Cold during execution but still configured to AP
  - + Predict if a state is Cold or Hot @ compile time using a small profiling input
  - + Propose topological-order based NFA partitioning into Predicted Cold and Predicted Hot states
  - + Develop SparseAP to handle mispredictions efficiently using our proposed Enable and Jump operations
- **Results**
  - + 2.1x Speedup (up to 47x)

We acknowledge the support of the National Science Foundation (NSF) grants (#1657336, #1717532, #1750667)