*ng*AP: Non-blocking Large-scale Automata Processing on GPUs

Session 9A: 12:00 PM, May 1
ASPLOS 2024

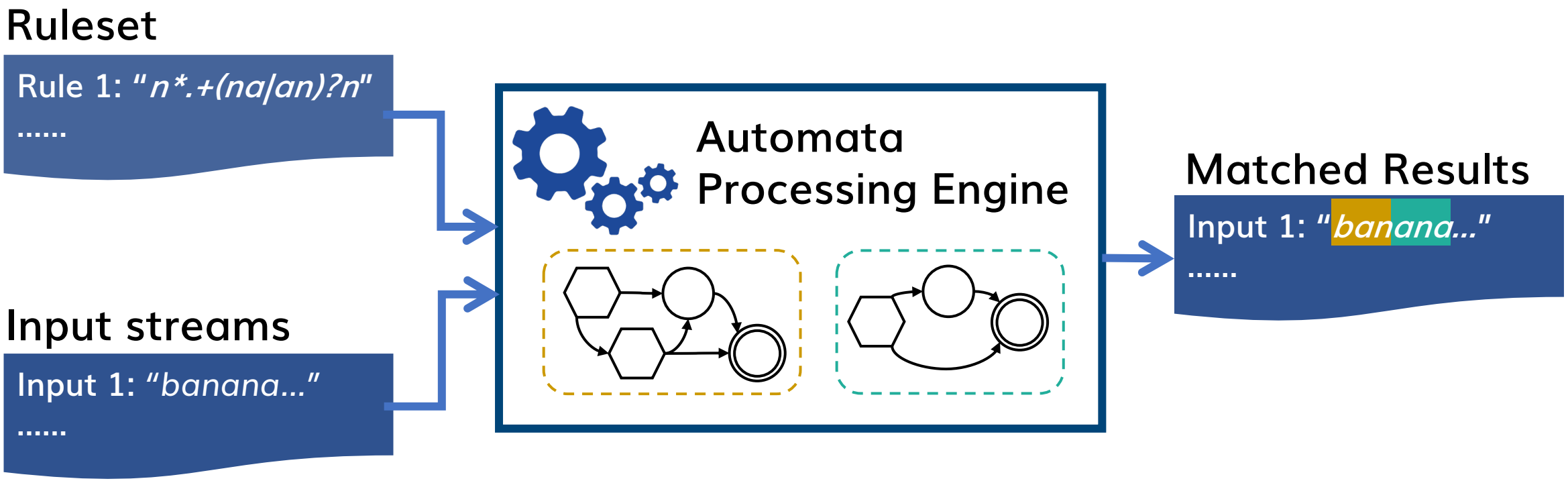
Tianao Ge[1], Tong Zhang[2], Hongyuan Liu[1]

[1]HKUST(GZ), [2]Samsung Electronics

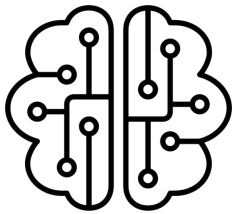
香港科技大學(廣州)
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

SAMSUNG

Automata Processing



Data Analytics



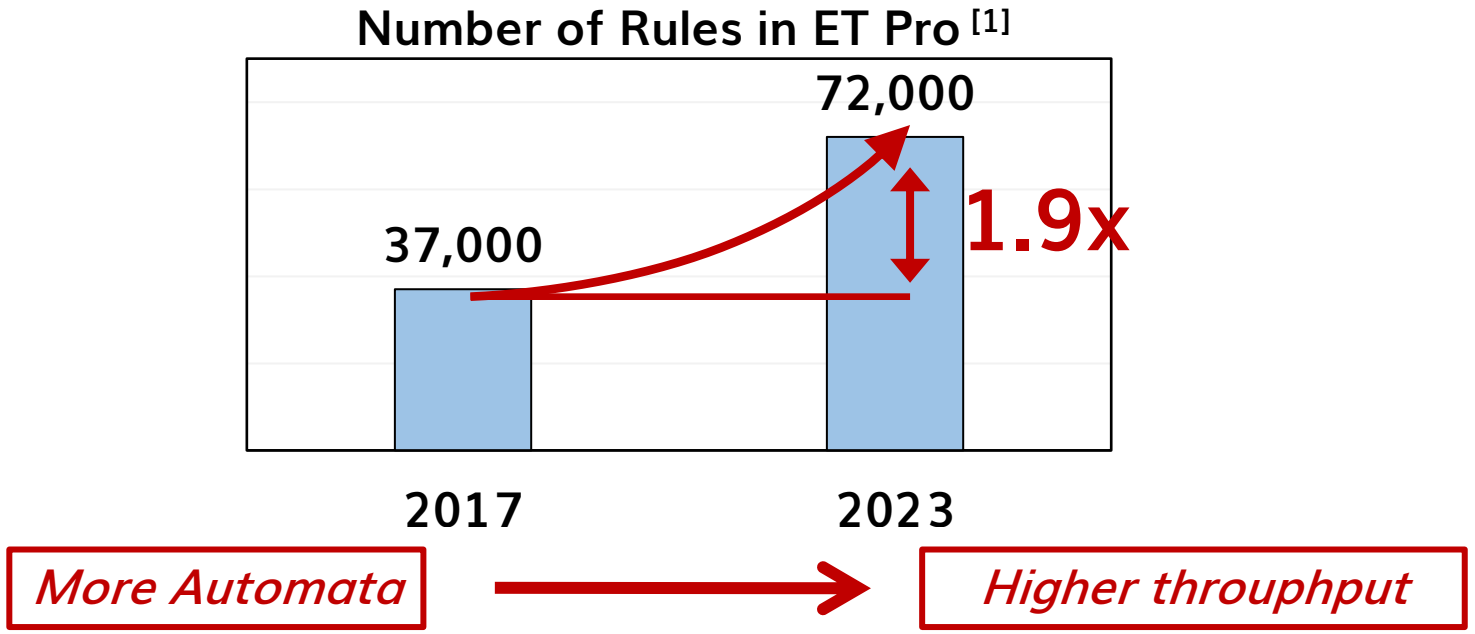
Machine Learning

Bioinformatics

Network Intrusion Detection

Automata are Scaling Up

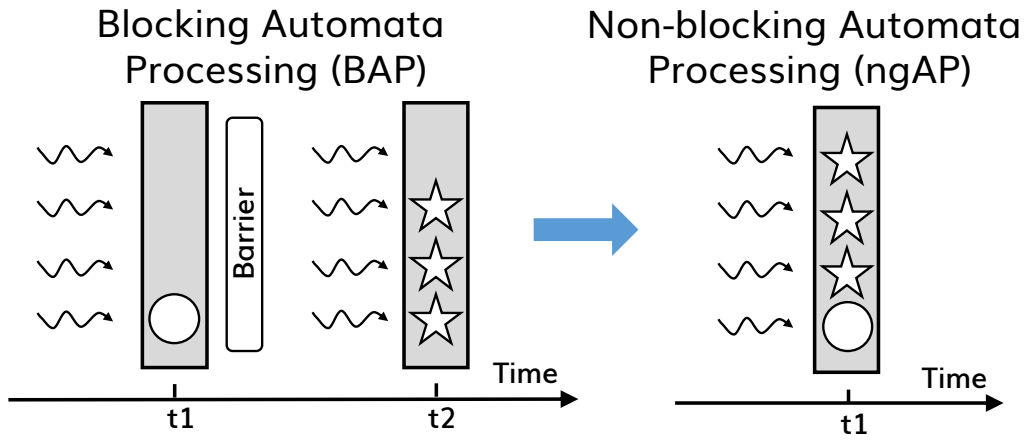
- In the network intrusion detection systems
  - the size of ruleset has increased by 90% from 2017 to 2023.
  - new rules are released everyday.

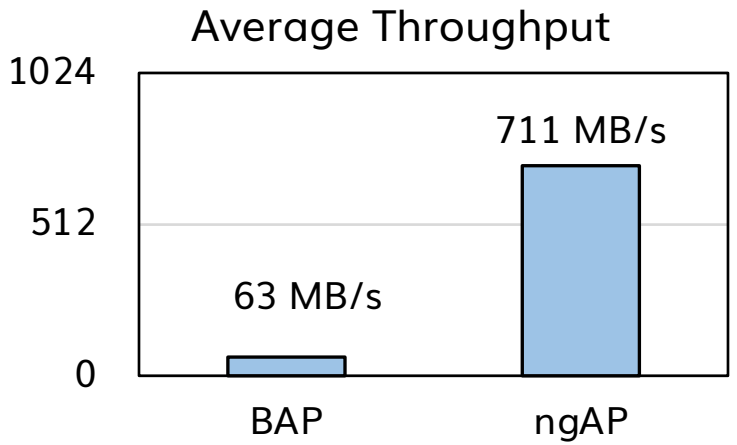


[1] https://www.proofpoint.com/us/threat-insight/et-pro-ruleset

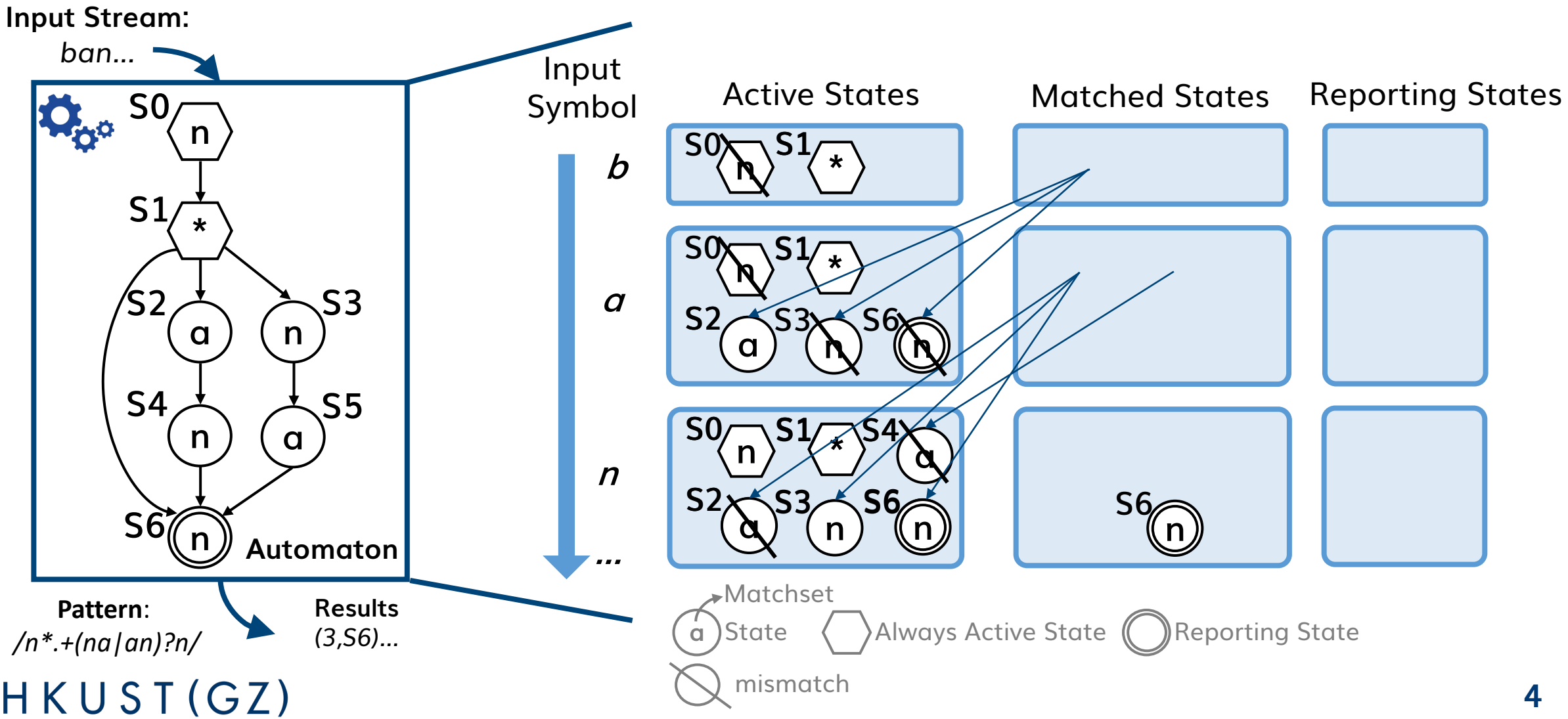
Outline

- Automata Overview
- Challenges on GPUs
  - GPU Threads Underutilization
  - Redundant Computations
  - Poor Data Locality
- *ng*AP: Non-blocking Automata Processing
- Evaluation

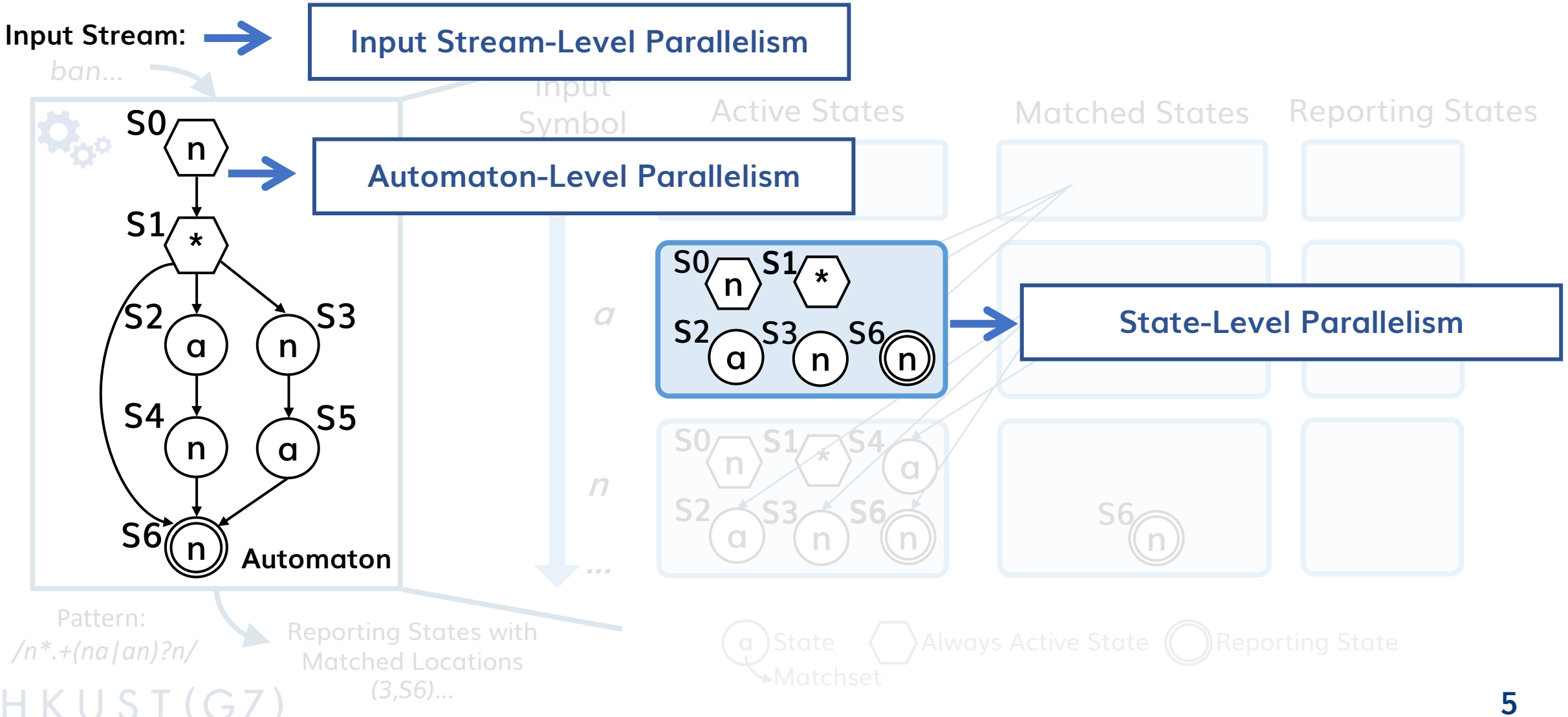




Parallelism in Automata Processing



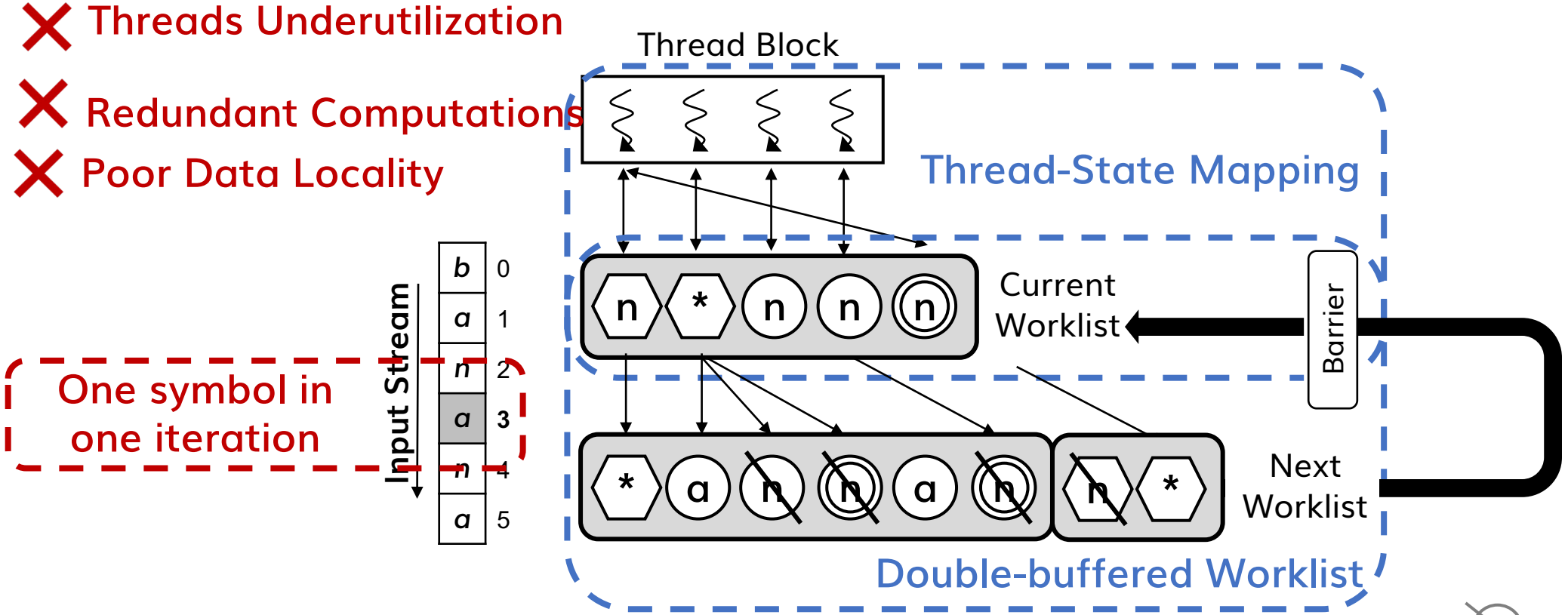
Parallelism in Automata Processing



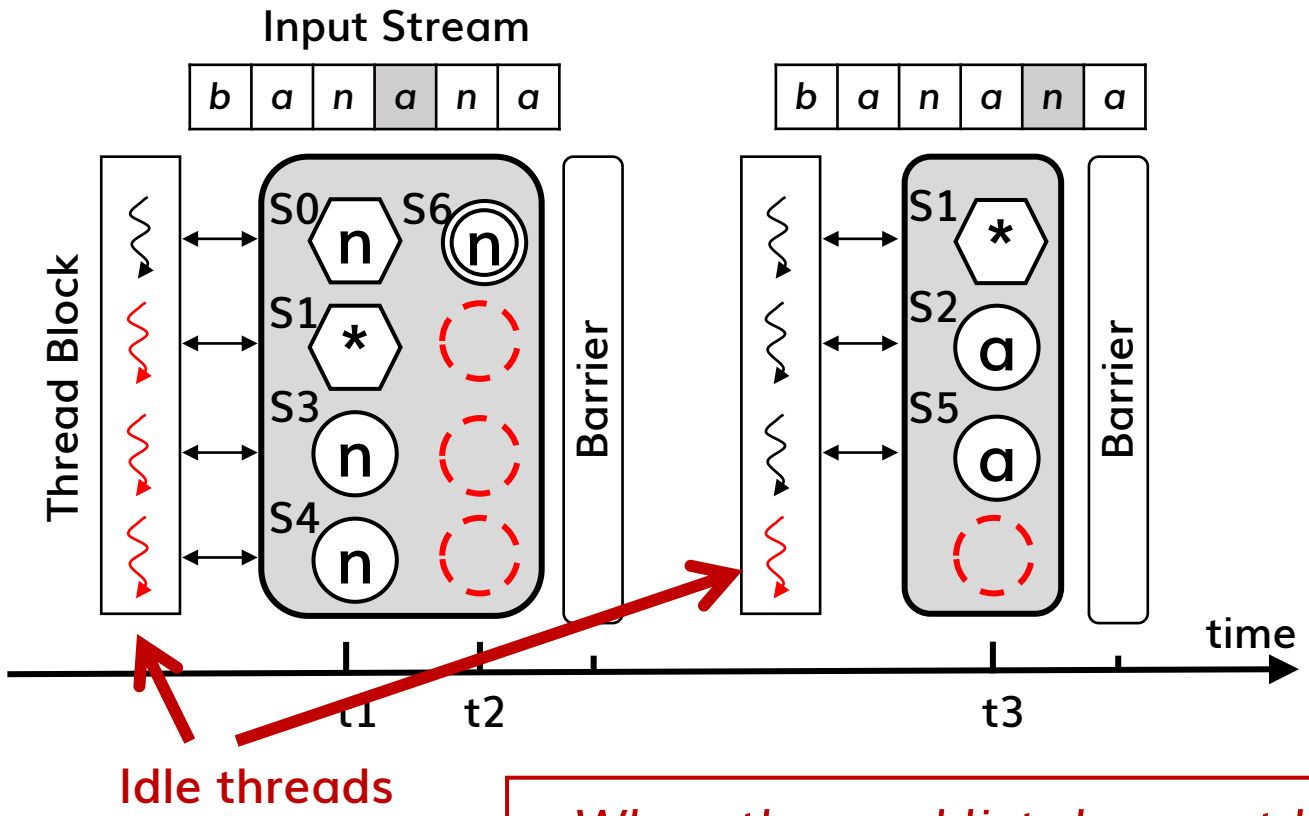
Prior Works on GPUs

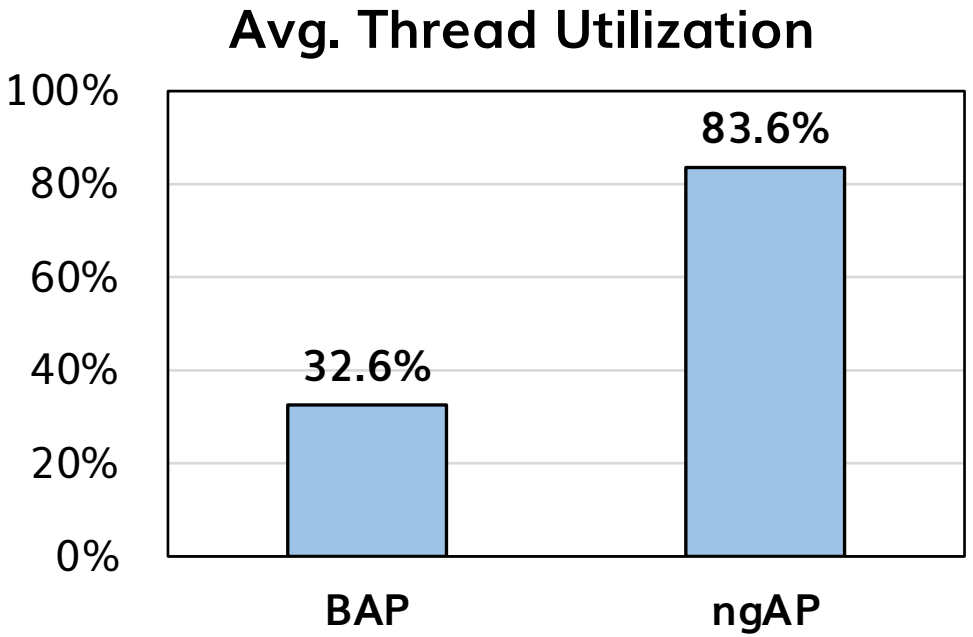
Blocking Automata Processing (BAP):

- ✗ Threads Underutilization
- ✗ Redundant Computations
- ✗ Poor Data Locality



Threads Underutilization

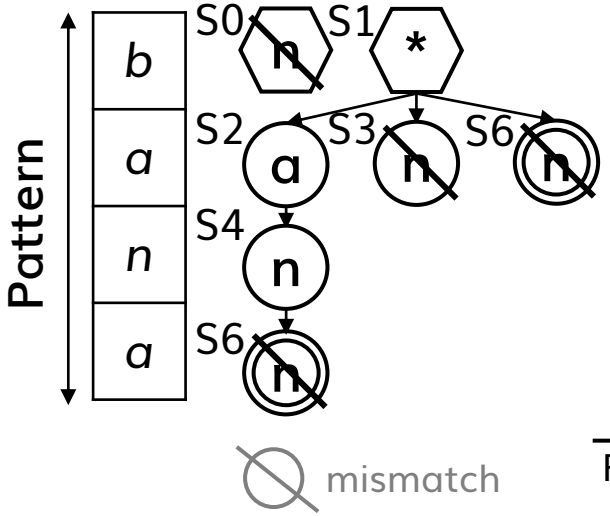




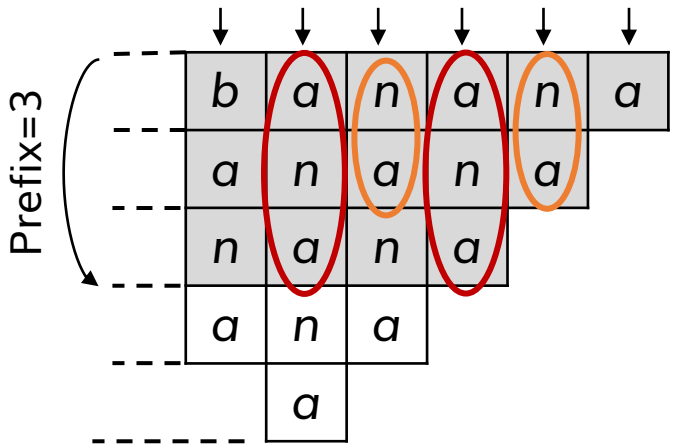
*When the worklist does not have enough states, the GPU threads are **underutilized**.*

Redundant Computations

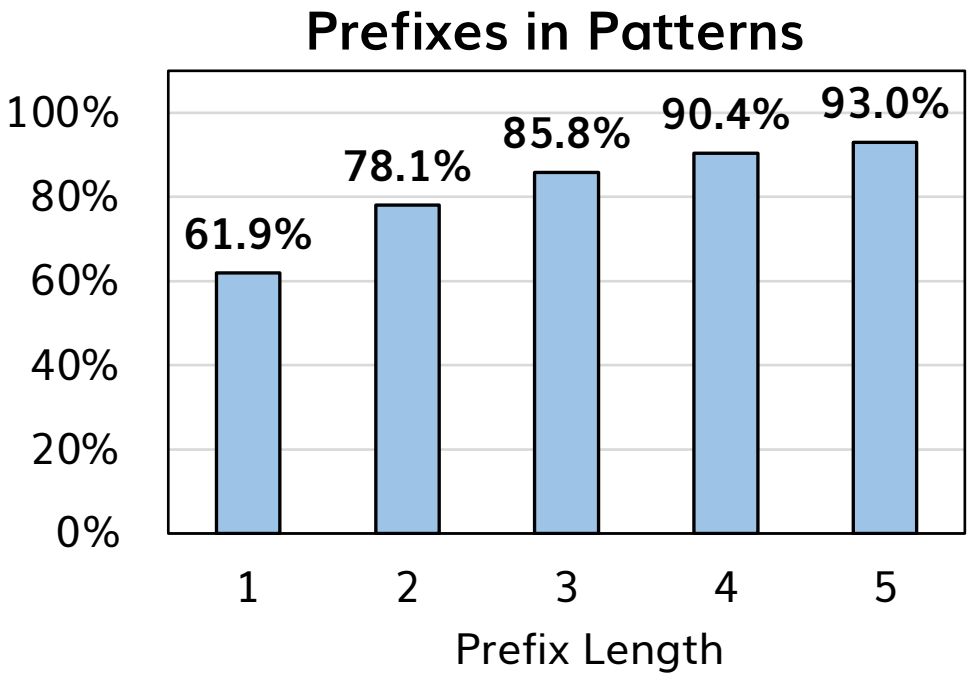
Matches begin with always-active states



Repeated Prefixes

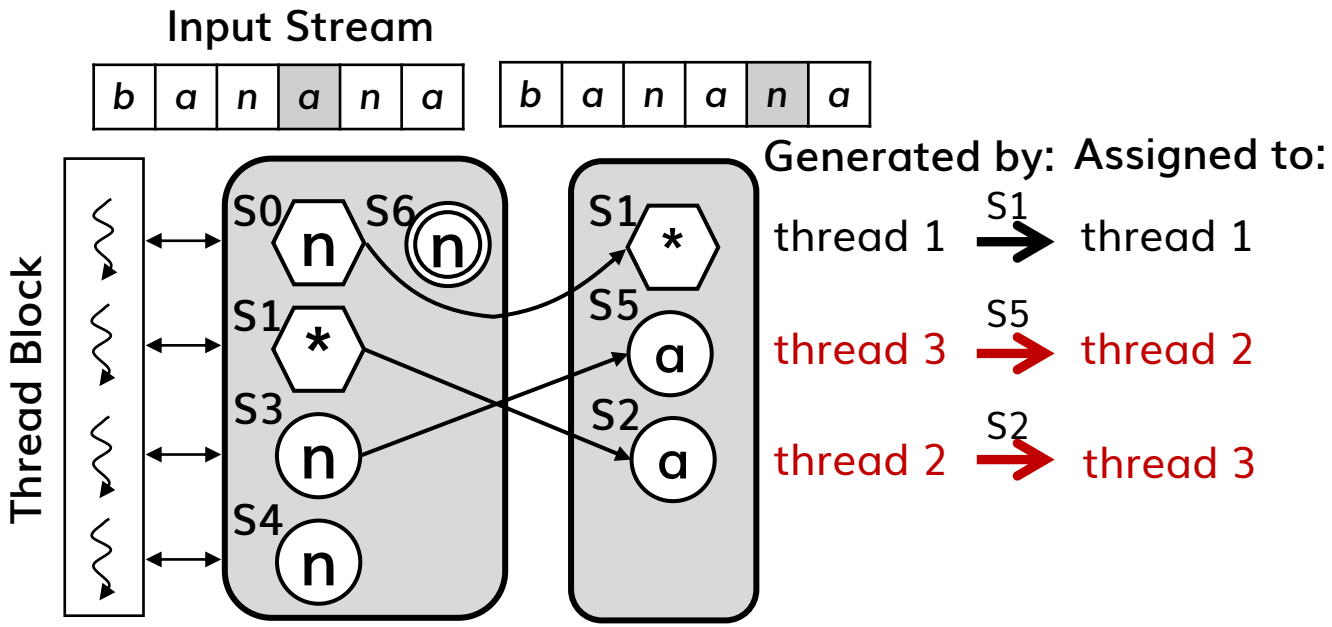


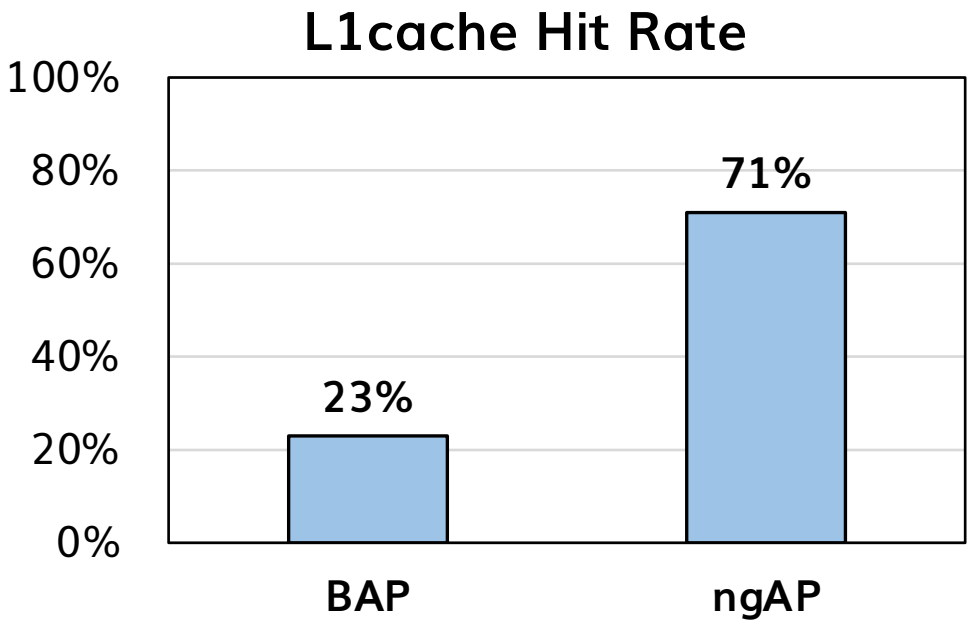
$$\frac{\text{Prefix Length}}{\text{Pattern Length}} = \frac{3+3+3+3+2+1}{4+5+4+3+2+1} \approx 79\%$$



*The repeated matches between pattern prefixes and always-active states are **redundant**.*

Poor Data Locality



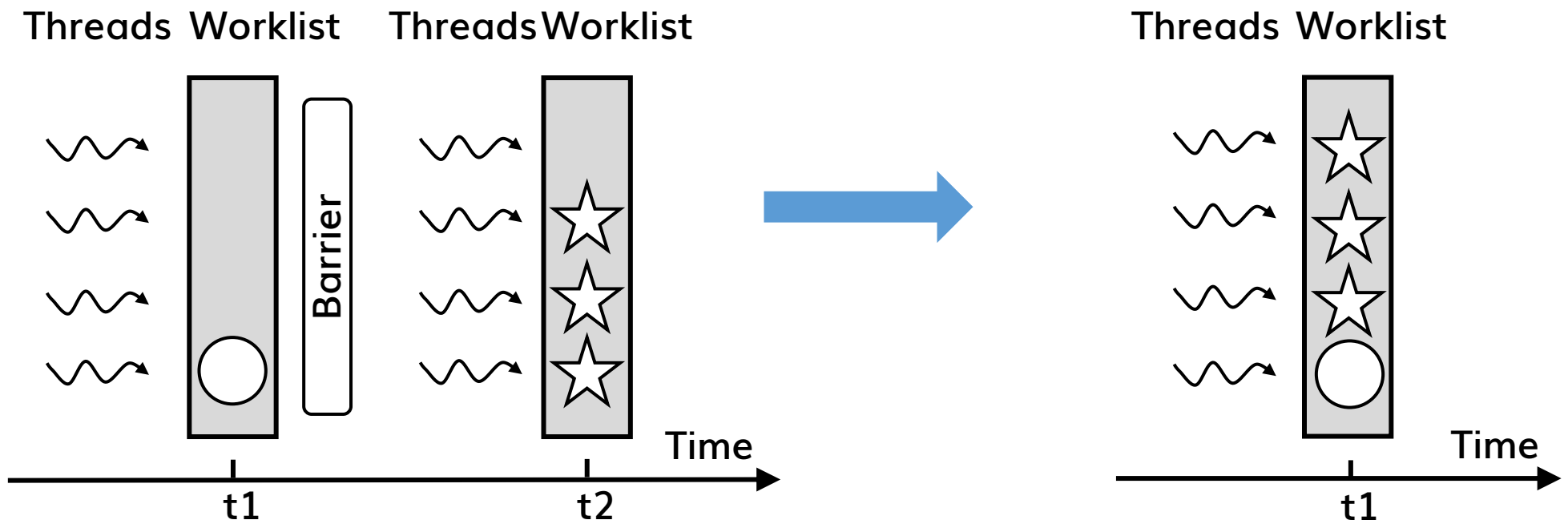


*The mapping between threads and states **switches frequently**.*

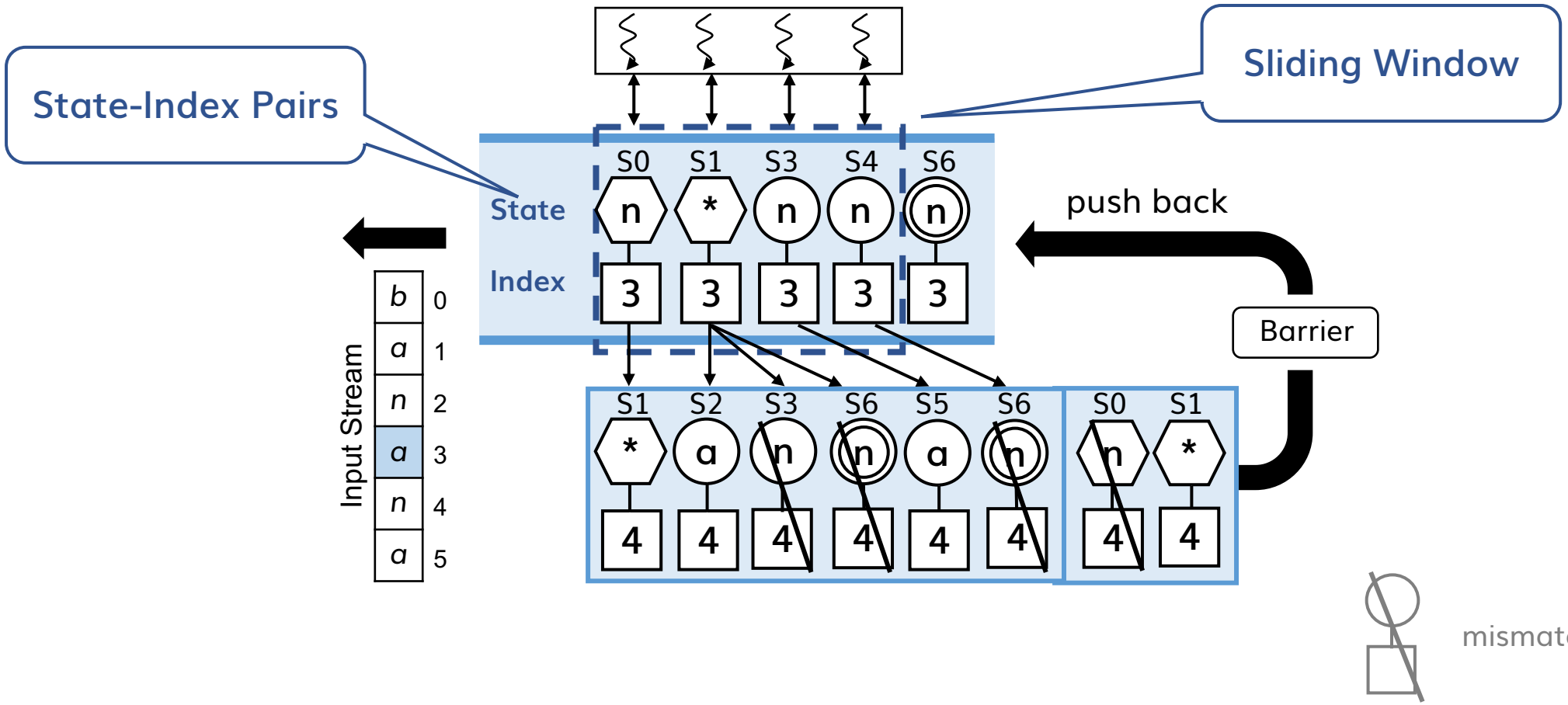
Key Idea: Non-blocking Processing

Blocking Automata Processing
(one symbol in one iteration)

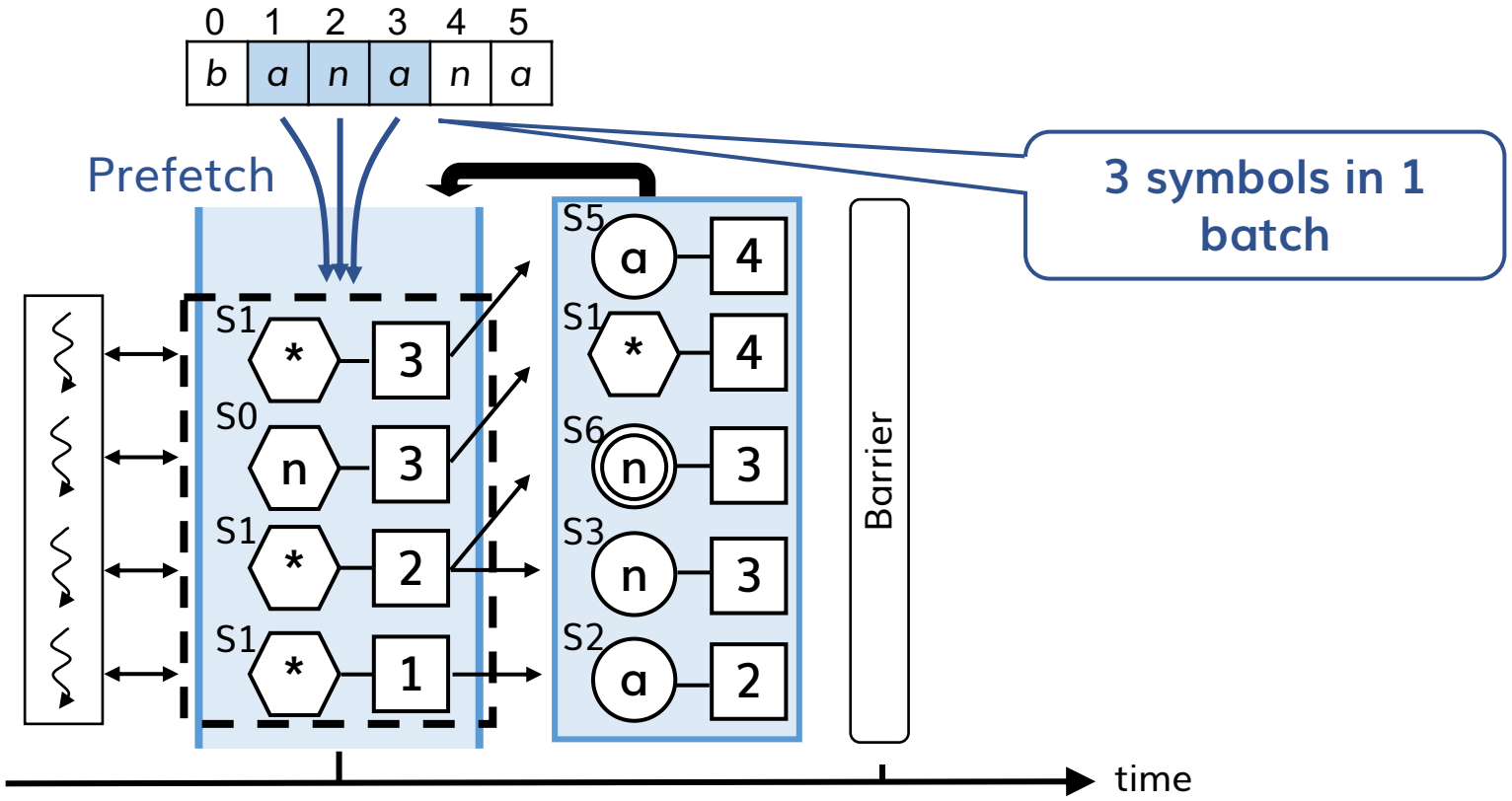
Non-blocking Automata Processing
(multiple symbol in one iteration)



Basic Design of *ng*AP

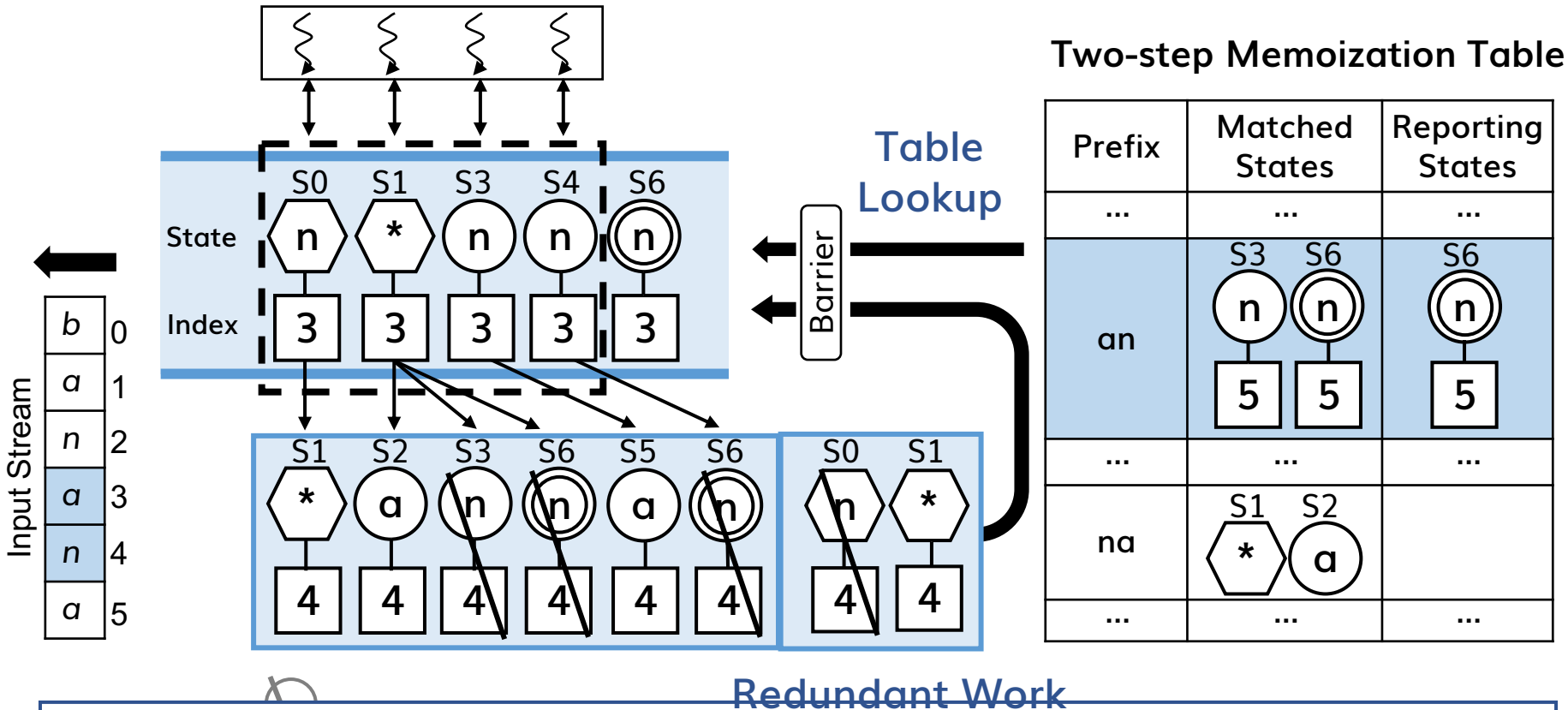


Opt#1 - Prefetching Always-Active States



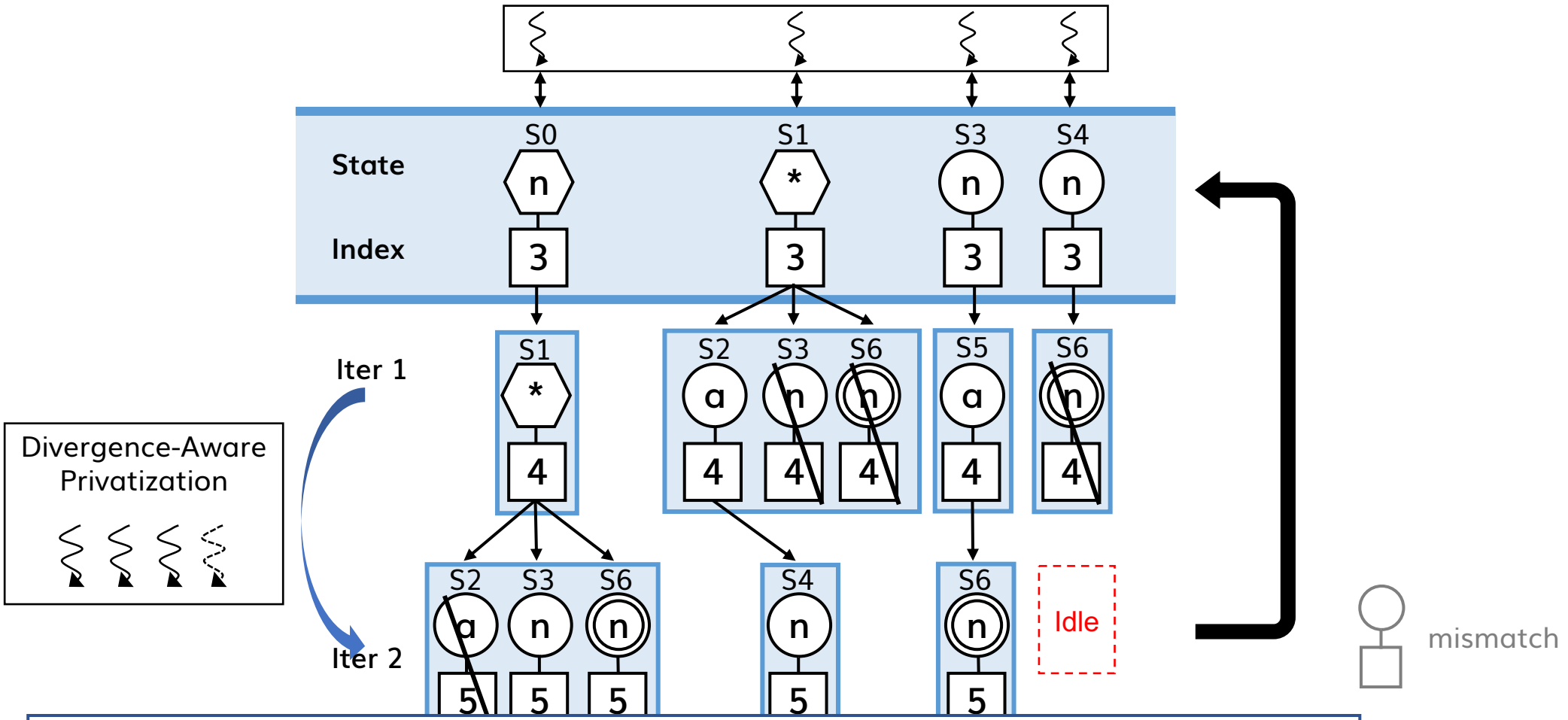
Prefetching significantly increase the number of elements coexisting in the worklist.

Opt#2 - Prefix Memoization



Redundant matches for prefixes are transformed into table lookups.

Opt#3 - Work Privatization

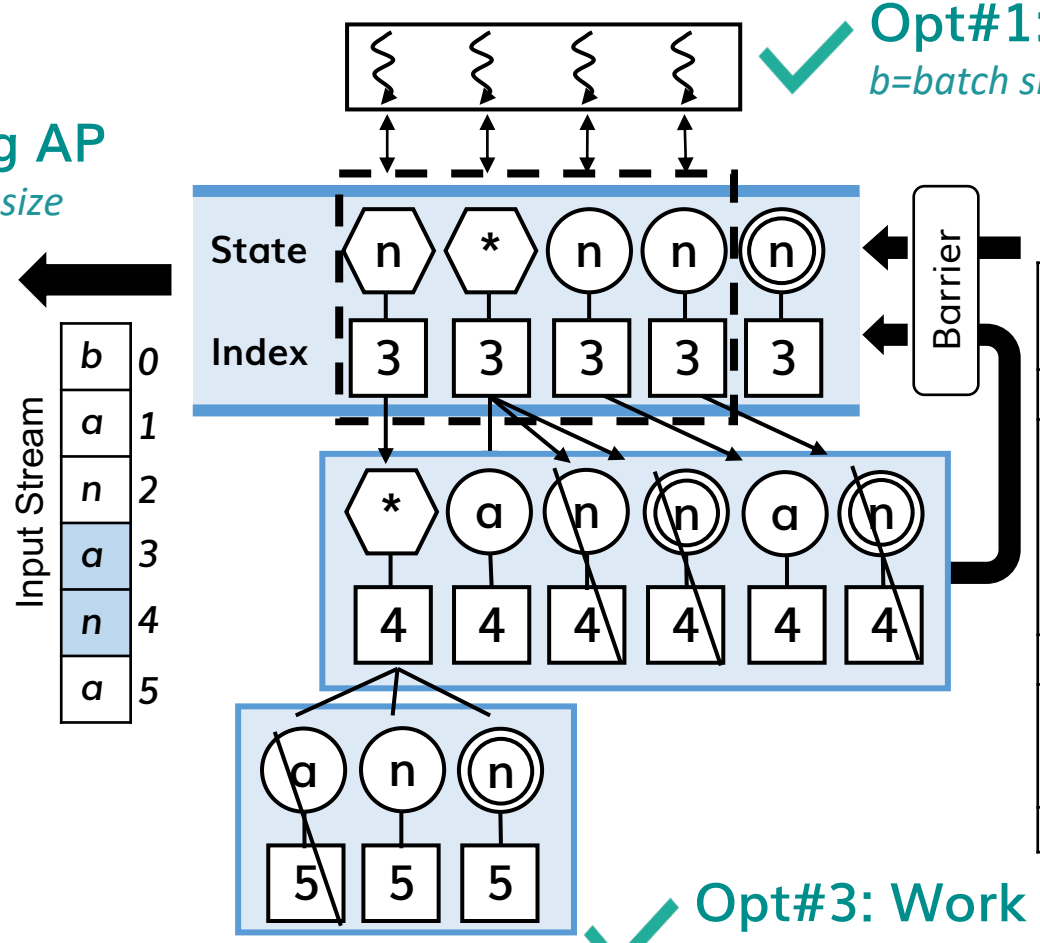


Work privatization retains the extended neighbors for threads without writing them back.

*ng*AP: Put it All Together

Non-blocking AP

s=sliding windows size



Opt#1: Prefetching Always-Active States

b=batch size

Opt#2: Prefix Memoization

p=prefix length

Memoization Table

Prefix	Matched States	Reporting States
...	...	...
an	n n (5 5)	n (5)
...	...	...
na	* a	
...	...	...

Opt#3: Work Privatization

d=divergense threshold

HKUST(GZ)

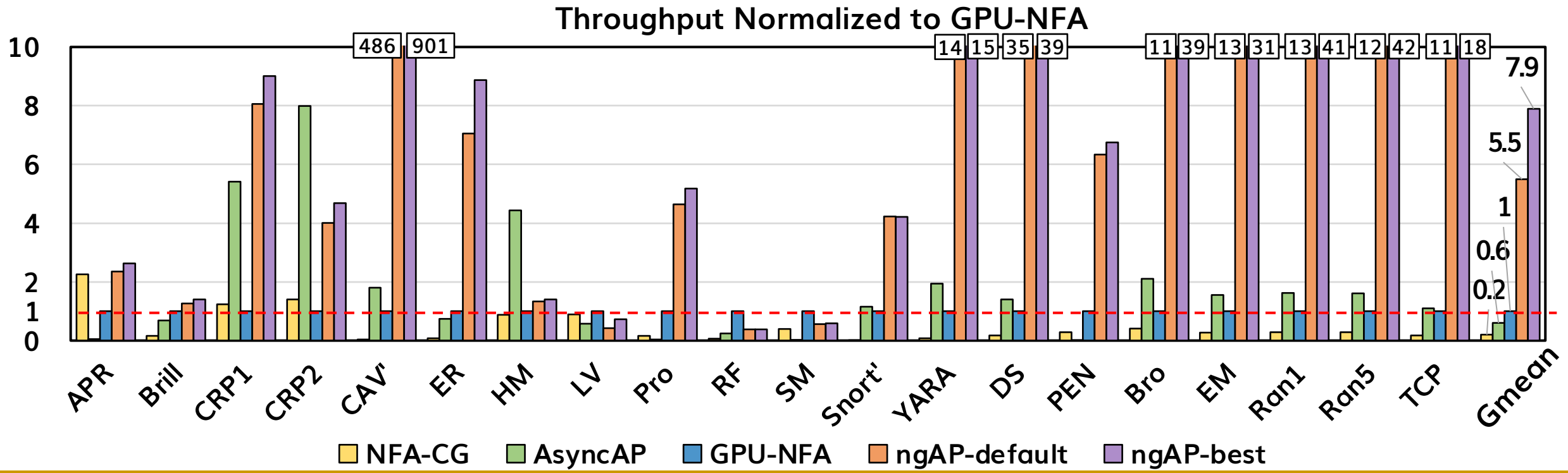
Methodology

- **Methods**
  - GPU
    - *ng*AP
      - Parameters: ngAP-default, ngAP-best
      - Optimizations: ngAP+O^1, ngAP+O^2, ngAP+O^3
    - NFA-CG [PPoPP'12]
    - GPU-NFA [ASPLOS'20]
    - AsyncAP [SIGMETRICS'23]
  - CPU
    - HyperScan [NSDI'2019]
- **Configuration**
  - NVIDIA RTX 3090
  - Intel Xeon 4214R CPU
  - 128 GB memory
  - GCC 9.5 and CUDA 12.0

- **Benchmarks**
  - 20 applications from *AutomataZoo* [IISWC'2018], *ANMLZoo* [IISWC'2016], and *Regex* [IISWC'2008]

Suite	Application	Abbr.
AutomataZoo	APPRNG4	APR
	Brill	Brill
	CRISPR_CasOFFinder	CRP1
	CRISPR_CasOT	CRP2
	ClamAV	CAV
	EntityResolution	ER
	Hamming_N1000_l18_d3	HM
	Levenshtein_l19d3	LV
	Protomata	Pro
	RandomForest_20_400_200	RF
	SeqMatch_BIBLE_w6_p6	SM
	Snort	Snort
	YARA	YARA
ANMLZoo	Dotstar	DS
	PowerEN	PEN
Regex	Bro217	Bro
	ExactMatch	EM
	Ranges1	Ran1
	Ranges05	Ran5
	TCP	TCP

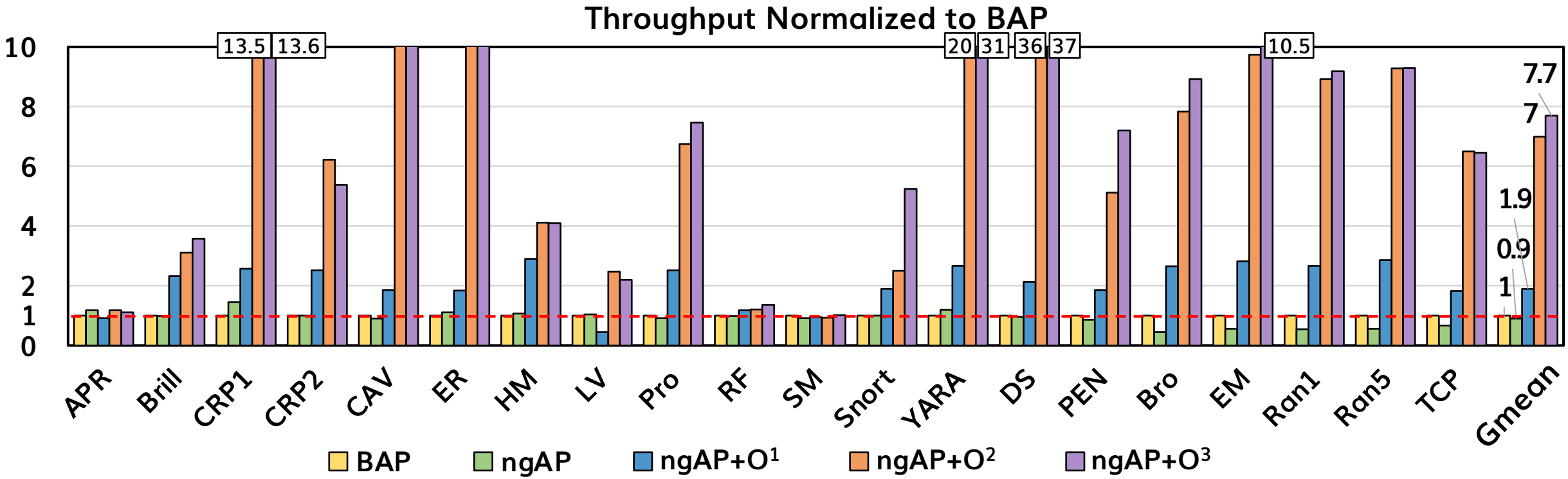
Evaluation - Overall Performance



Compared GPU baseline, *ng*AP achieves an average speedup of 7.9×, with a peak of up to 901×, across 20 applications.

Compared CPU baseline (*Hyperscan*), *ng*AP achieves an average speedup of 11.5×.

Evaluation - Performance Breakdown



The three optimizations based on ngAP significantly improve the performance by 1.9x, 7x and 7.7x.

Conclusion

- Key Insight: “one-symbol-at-a-time” serializes the execution!
- ***ngAP***: Non-blocking Automata Processing
  - Prefetching Always-Active States
  - Prefix Memoization
  - Work Privatization
- **7.9×** to **901×** throughput speedup

https://github.com/getianao/ngAP

Tianao Ge (tge601@connect.hkust-gz.edu.cn),
Tong Zhang, Hongyuan Liu

香港科技大學(廣州)
THE HONG KONG
UNIVERSITY OF SCIENCE AND
TECHNOLOGY (GUANGZHOU)

SAMSUNG