ASSOCIATION RULE MINING WITH MICRON AUTOMATA PROCESSOR

Ke Wang[1,2], Yanjun Qi[1,2], Jeffrey J. Fox[1,3],
Mircea R. Stan[1,4], Kevin Skadron[1,2]

[1] *Center for Automata Computing*

[2] *Department of Computer Science*

[3] *Department of Material Science*

[4] *Department of Electrical and Computer Engineering*

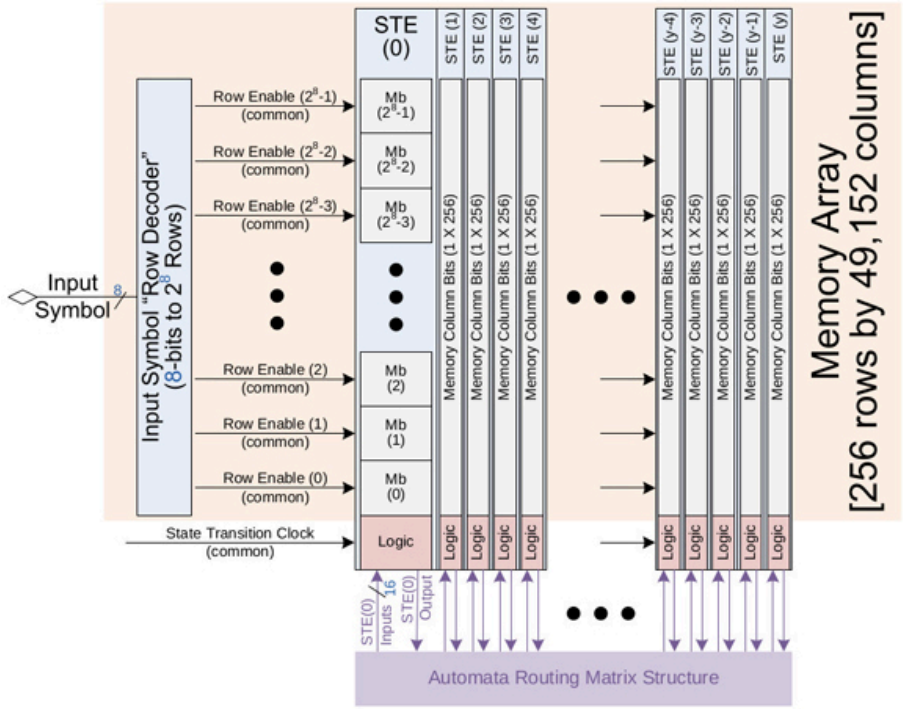
University of Virginia

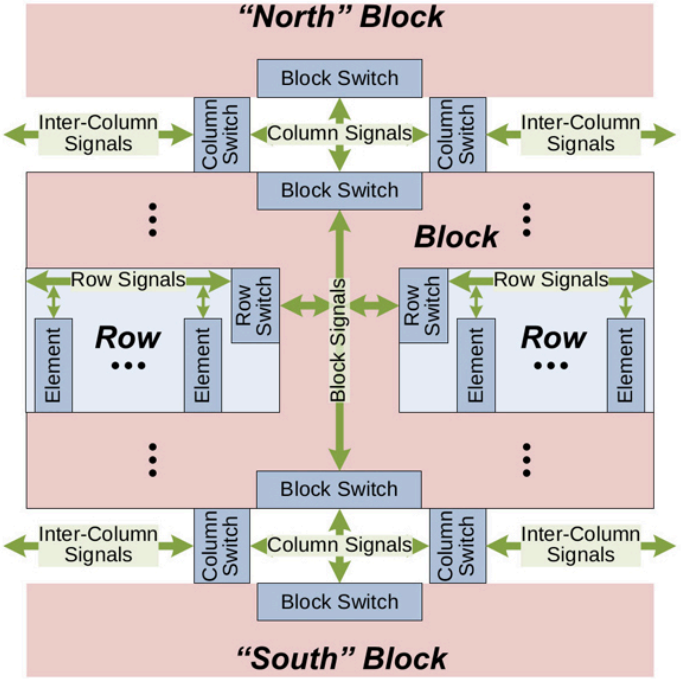
Outline

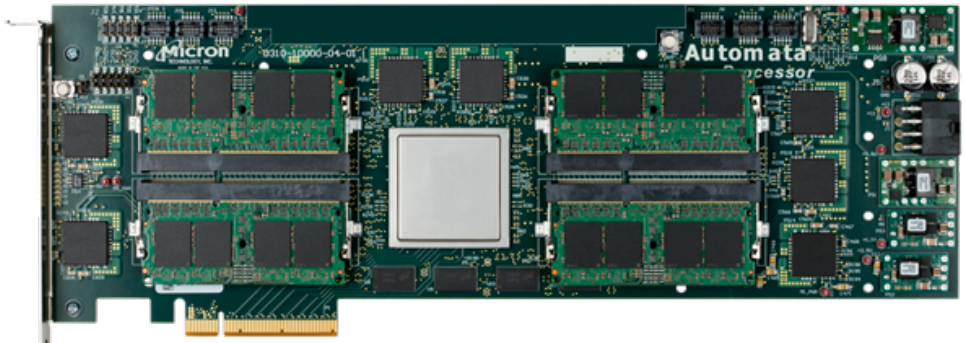
- Micron Automata Processor (AP)
- Association Rule Mining (ARM)
- ARM on AP: Opportunities and Challenges
- AP Accelerated ARM and optimizations
- Performance Evaluation
- Conclusions and the Future Work

Micron Automata Processor

❑ Architecture

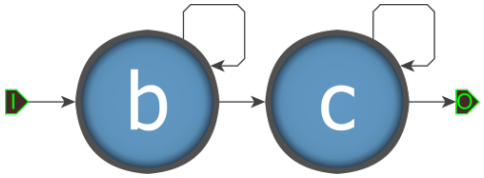


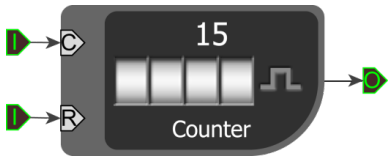


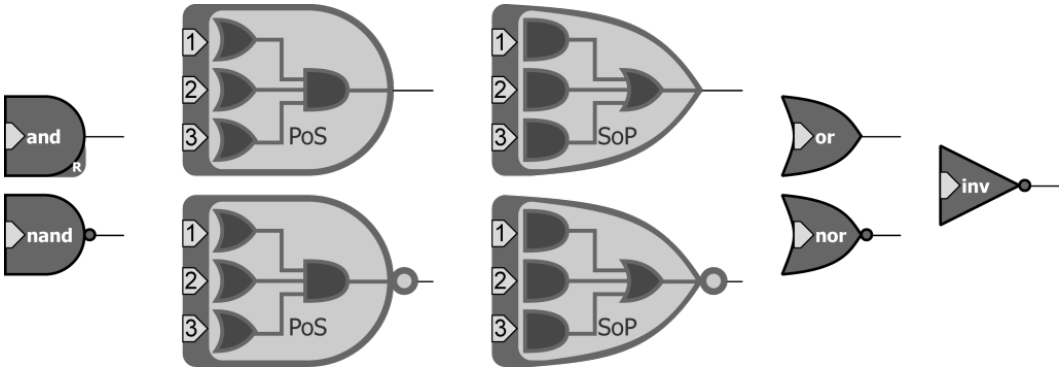


❑ Function Elements and Capacity

	per chip
State Transition Element (STE)	49,152
Counter Element	768
Boolean Logic Element	2,304







32-48 chips/board -> 1.5-2.5 million concurrent operations

Micron Automata Processor

❑ Input and Output

- The AP chip process one 8-bit symbol each cycle
- *Multiple-Instruction Single-Data (MISD) architecture* (Flynn's "dark" corner)
- Each AP chip can process up to 6 separate data streams concurrently
- Any STE can be configured to accept the first symbol in the stream (start-of-data mode), or every symbol in the input stream (all-input mode)

❑ Programming and Reconfiguration

- Automata Network Markup Language (ANML) is an XML language for describing the composition of automata networks
- ANML support the feature of macro, a container of automata for functional encapsulating
- Bundling with other languages: C, Python and Java
- GUI developing environment: AP Workbench
- Fast reconfiguration: 50ms for whole board, 45ms for symbol replacement only

Association Rule Mining

Association rule mining (ARM, or frequent itemset mining, FIM):

- Identify ***strong rules*** discovered in databases
- The order of items within a transaction doesn't matter

- Web usage mining
- Traffic accident analysis
- Intrusion detection

- Market basket analysis
- Bioinformatics

Trans.	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer, Coke
5	Bread, Milk, Coke, Diaper

Itemset
K-Itemset
Support: number of transactions which contain this itemset
sup({Diaper, Milk})= 3
Minimum Support: threshold to tell frequent or not

ARM on AP: Opportunities and Challenges

❑ Opportunities

- The basic operation of ARM is pattern matching and counting
- The huge capacity of STE and counter elements of AP allows matching/counting hundreds – thousands itemsets in parallel

❑ Challenges

- The patterns in ARM are discontinuous

  For set {a,m}, the following transactions are all matched

  {a,m}, {a,b,m,}, {a, c, d, m, l}

- The patterns in ARM are unordered

  For set {a,m}, the following transactions are all matched

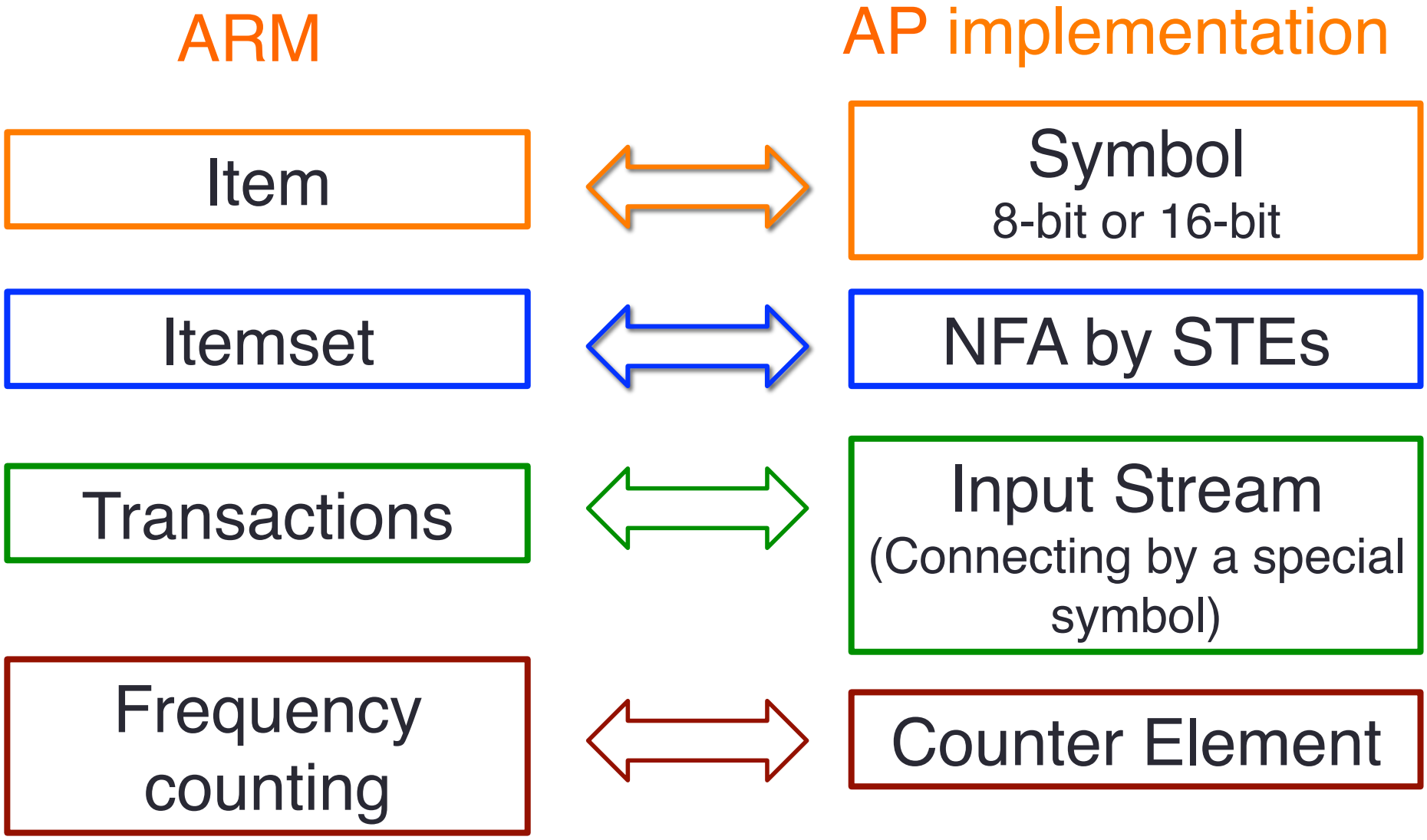
  {a,m}, {m, b, a}, {m, a}

- The number of patterns in ARM grow exponentially with the size of itemset

AP Accelerated ARM - Algorithm

- Apriori framework:
  - Downward-closure property: a frequent itemset, all its subsets are also frequent and thus for an infrequent itemset, all its supersets must also be infrequent
- Apriori Algorithm:
  - Candidates of frequent (K+1)-itemsets are generated from K-itemsets
  - Count the frequencies of candidates one by one to determine the frequent ones
  - From 1 to n level itemset mining
  - AP is used to accelerate each level

AP Accelerated ARM – Concept



AP Accelerated ARM - Flowchart

Data preprocessing:

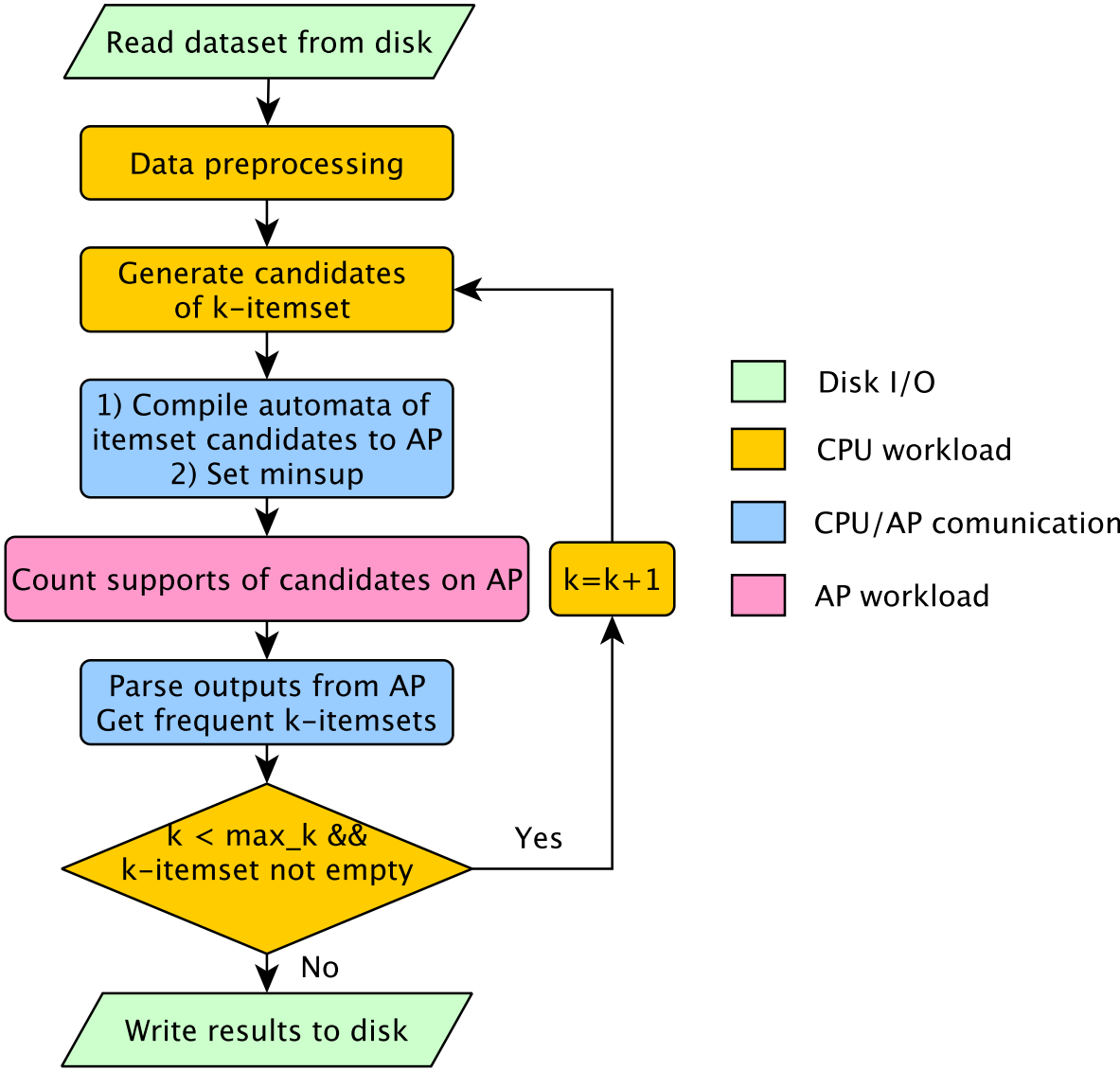
1) Filter out infrequent items
2) Recode -> 8-bit / 16-bit symbols
3) Recode transactions
4) Sort items in transactions
5) Connect transactions by a special symbol (\x255)

Encoding:

freq_item# <255: 8-bit
254< freq_item# <64516:16-bit

Sorting:

Descending sorting according to item frequency [1]



[1] Christian Borgelt, "Efficient implementations of Apriori and Eclat," in Proc. FIMI '03 , 2003

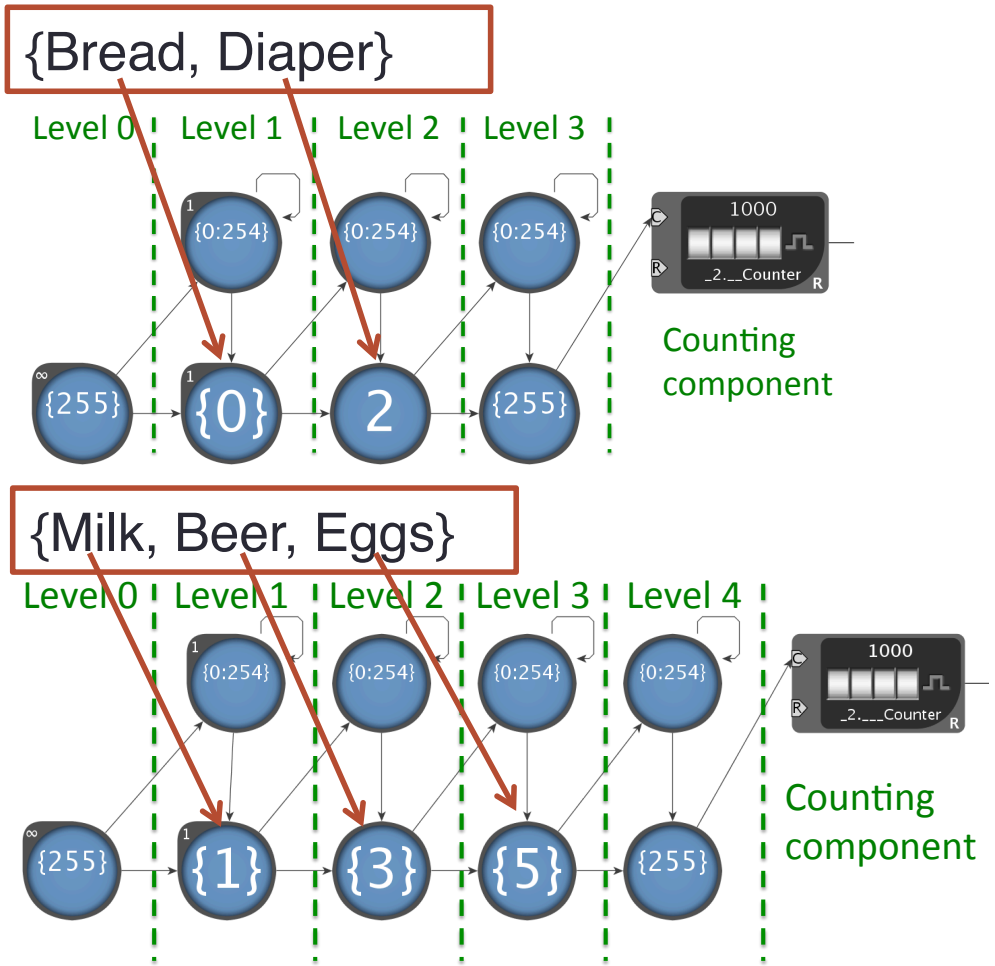
AP Accelerated ARM – Automata Design

Trans.	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread,Milk, Diaper, Beer, Coke
5	Bread, Milk, Diaper, Coke

Item	Code
Bread	0
Milk	1
Diaper	2
Beer	3
Coke	4
Eggs	5
Separator	255(\xFF)

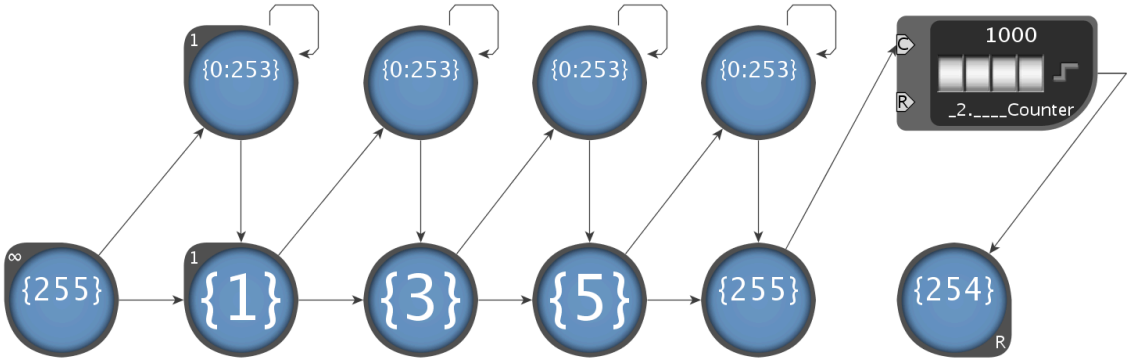
Transaction stream:

01\xFF0235\xFF1234\xFF01234\xFF0124



AP Accelerated ARM – Optimization

❑ I/O minimization



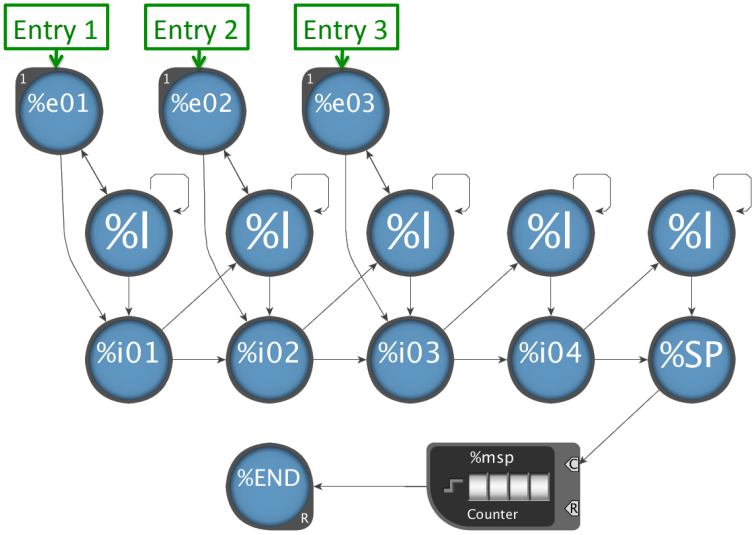
- Add one more STE with a special symbol “254” for the ending sign of input
- The counter keep activating this STE after its threshold is reached
- Add one more ending symbol “254” in the end of input stream
- Only last cycle gets reports; avoid multiple report vectors during processing

❑ Concurrent mining k-and (k+1)-itemsets

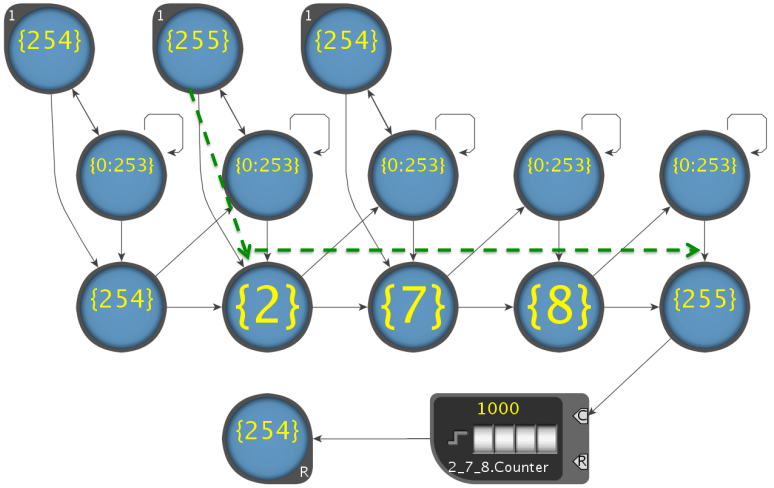
- If capacity allows, mine k-itemsets and (k+1)-itemsets in parallel
- To generate (k+1)-itemset candidates, assume all k-itemset candidates are frequent

❑ Avoid routing reconfiguration:

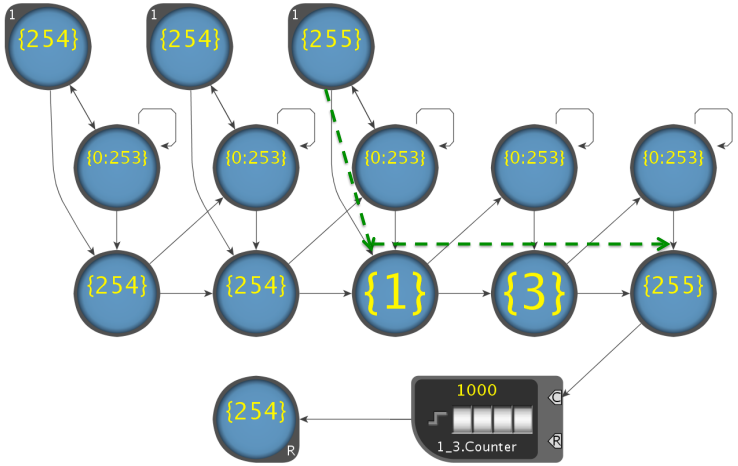
Multiple-entry NFA for variable-size itemset (MENFA-VSI).



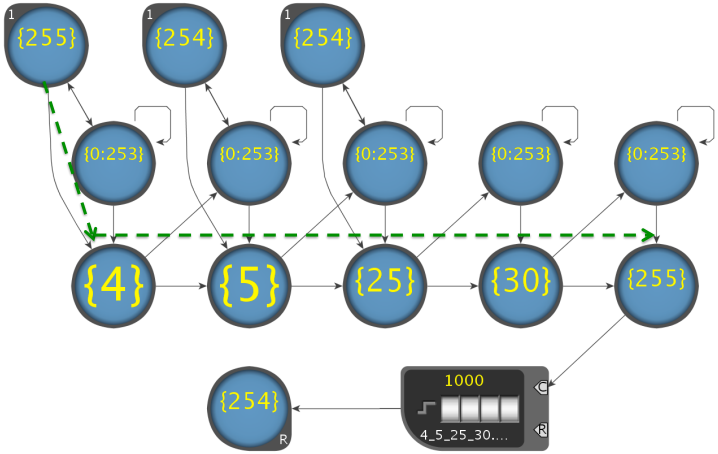
(a) AP macro of ME-NFA-VSI



(c) Automaton for itemset {2, 7, 8}



(b) Automaton for itemset {1, 3}



(d) Automaton for itemset {4,5,25,30}

Performance Evaluation – AP

❑ AP capacity and overhead

- Target hardware: D480 X 48
- 8-bit encoding:
  - Capacity: 384 ME-NFA-VSI /chip, 18432/board
  - Speed: 7.5ns/ item
  - A ME-NFA-VSI supports 2-itemset to 40-itemset
- 16-bit encoding:
  - Capacity: 384 ME-NFA-VSI /chip, 18432/board
  - Speed: 15ns/item
  - A ME-NFA-VSI supports 2-itemset to 24-itemset
- Symbol replacement time: 45ms for an entire board

Performance Evaluation - Comparison

❑ Compare with other implementations

1. Borgelt's Apriori CPU sequential implementation [1] : Apriori-CPU
2. A CPU serial implementation of Equivalent Class Transformation (Eclat) [2]: Eclat-1C
3. A CPU multi-threading implementation of Eclat[2] : Eclat-6C
4. A GPU implementation of Eclat[2]: Eclat-1G

❑ Testing platform

- CPU: Intel(R) Xeon(R) CPU E5-1650(6 physical cores 3.20GHz)
- Mem: 32GB, 1.333GHz
- GPU: Nvidia Kepler K20C, 706 MHz clock, 2496 CUDA cores, 4.8GB global memory

[1] C. Borgelt, "Efficient implementations of apriori and eclat," in Proc. FIMI '03 , 2003, p. 90.'
[2] F. Zhang, Y. Zhang, and J. D. Bakos, "Accelerating frequent itemset mining on graphics processing units," J. Supercomput., vol. 66, no. 1, pp. 94–117, 2013.

Performance Evaluation - Datasets

- Four real-world datasets

Table I: **Real-World Datasets**

Name	Trans#	Aver. Len.	Item#	Size (MB)
Pumsb	49046	74	2113	16
Accidents	340183	33.8	468	34
Webdocs	1692082	177.2	5267656	1434
ENWiki	11507383	70.3	6322092	2997.5

Pumsb, Accidents and Webdocs are from *Frequent itemset mining dataset repository*," http://fimi.ua.ac.be/data/.
ENWiki was generated English Wikipedia 2014

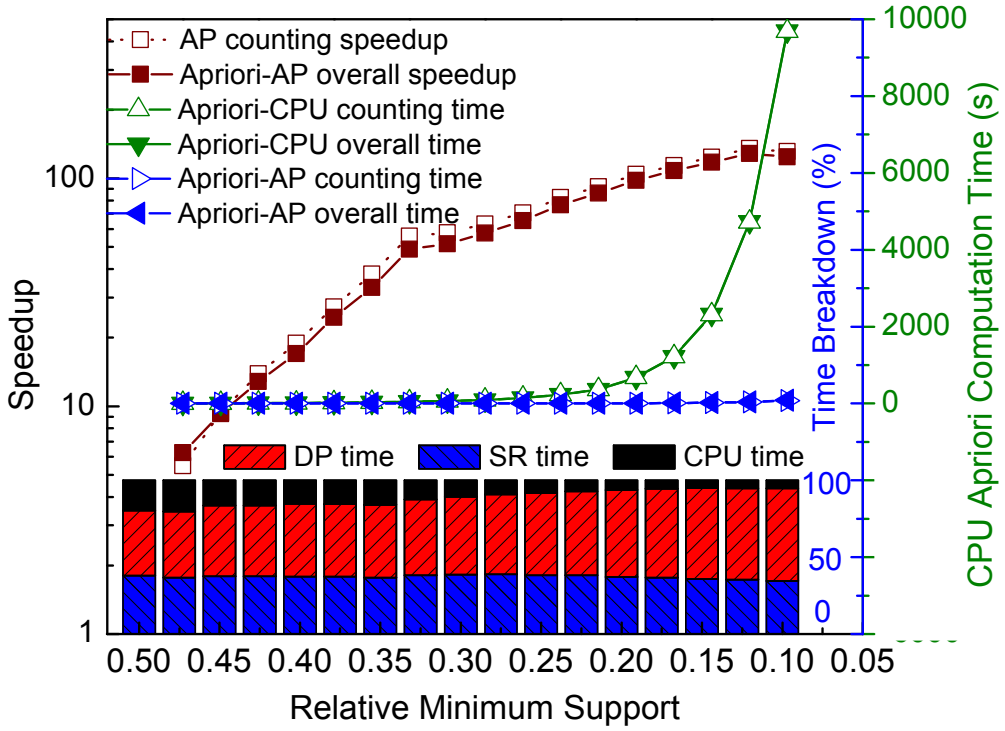
- Three synthetic datasets

Table II: **Synthetic Datasets**

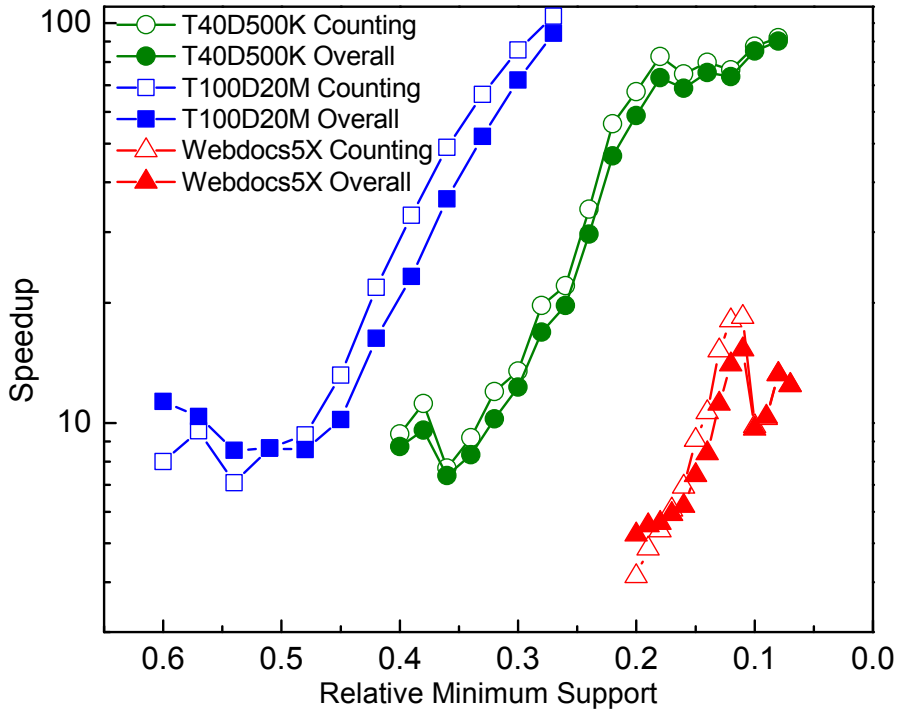
Name	Trans#	Aver. Len.	Item#	ALMP	Size (MB)
T40D500K	500K	40	100	15	49
T100D20M	20M	100	200	25	6348.8
Webdocs5X	8460410	177.2	5267656	N/A	7168

T40D500K and *T100D20M* ware generated from IBM Market-Basket Synthetic Data Generator
Webdocs5X is generated by duplicating transactions of Webdocs 5 times

Performance Evaluation – vs. Apriori-CPU

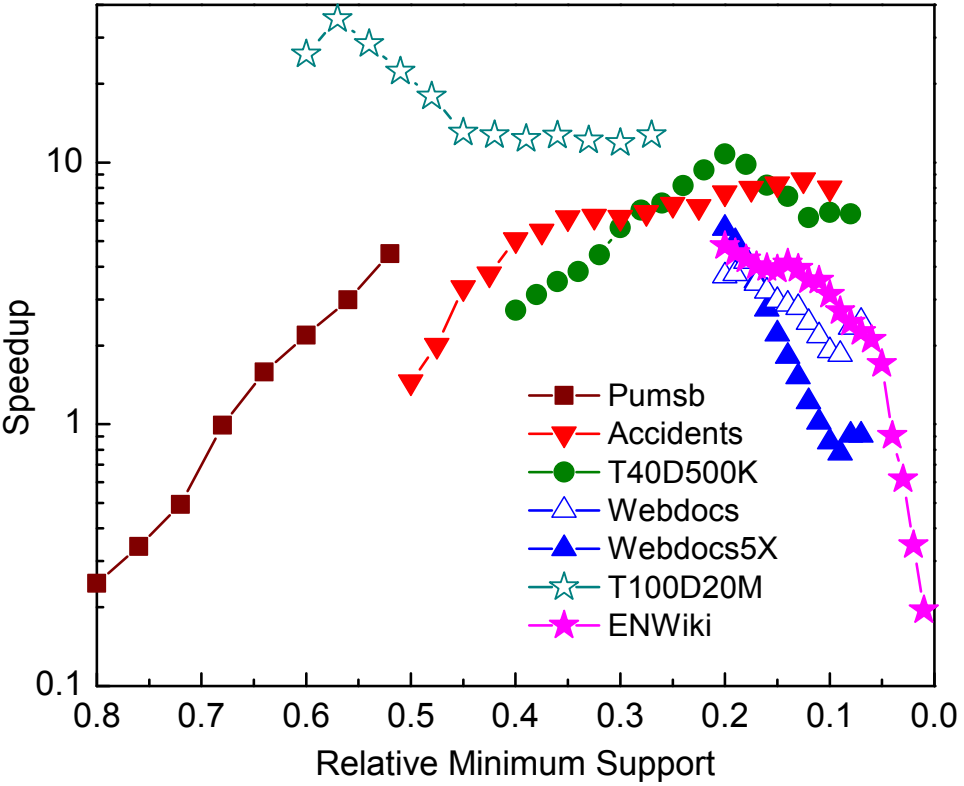


The performance results of Apriori-AP on Accidents. DP time, SR time and CPU time represent the data process time on AP, symbol replacement time on AP and CPU time respectively。



The speedup of Apriori-AP over Apriori-CPU on three synthetic benchmarks

Performance Evaluation – Apriori vs. Eclat



The performance comparison of CPU sequential Apriori and Eclat

Thought Eclat (Equivalent Class Clustering) has better performance on CPU, it is not a good fit for the AP:

1) Eclat requires bit-level operations; AP works on byte-level symbols

2) Eclat updates vertical representations of transactions for each new itemset candidate; dynamically changing the input stream is not efficient using the AP

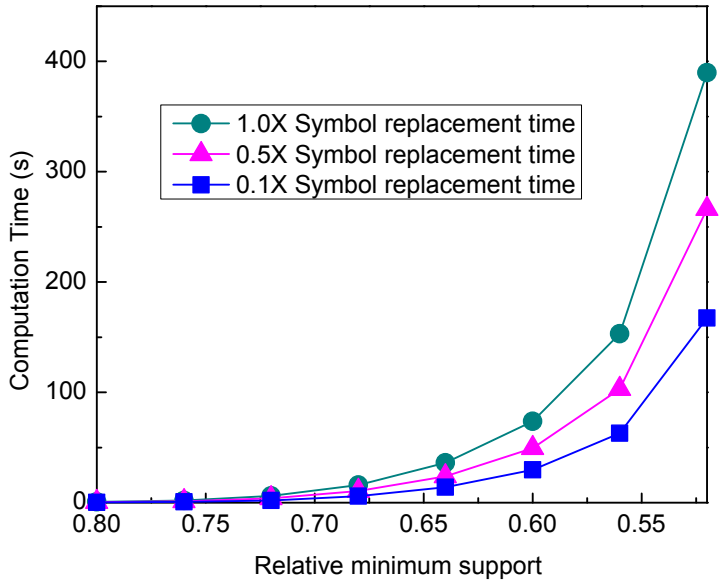
3) Even the hybrid search strategy cannot expose enough parallelism to make full use of the AP chips

Though Eclat has 8X performance advantage in average cases, the vertical bitset representation become less efficient for sparse and large dataset (high #trans and #freq item ratio).

Performance Evaluation – Architecture impacts

❑ STE symbol replacement time

The symbol replacement latency can be quite important for small and dense datasets that require multiple passes in each Apriori iteration, but this latency may be significantly reduced in future generations of the AP.



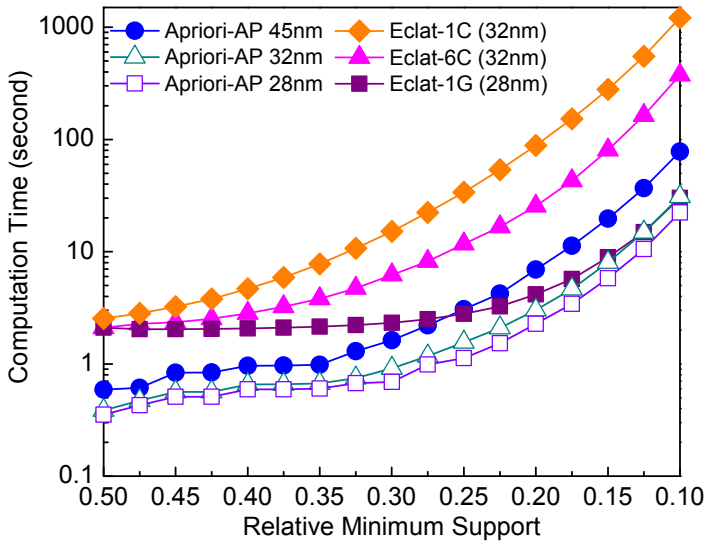
The impact of symbol replacement time on Apriori-AP performance for Pumsb

❑ Normalizing for technology

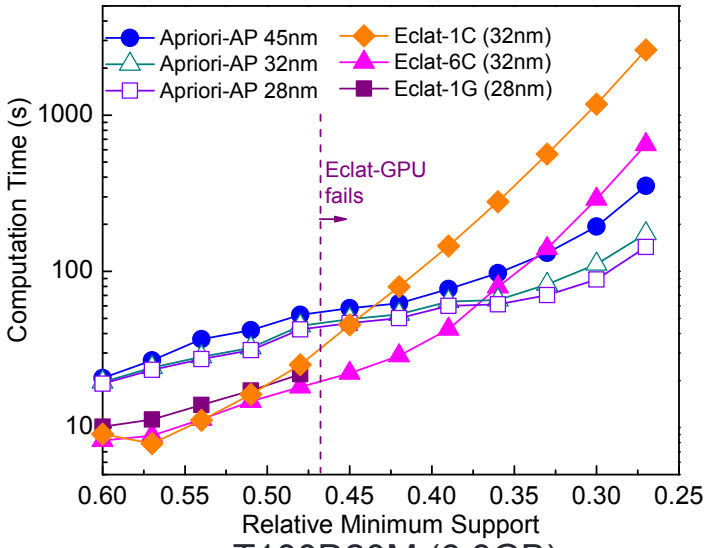
To compare the different architectures in the same semiconductor technology mode, we show the performance of technology projections on 32nm and 28nm technologies assuming linear scaling for clock frequency and square scaling for capacity[1].

[1] J. Rabaey, A. Chandrakasan, and B. Nikoli´c, Digital Integrated Circuits , 2nd ed. Pearson Education, 2003.

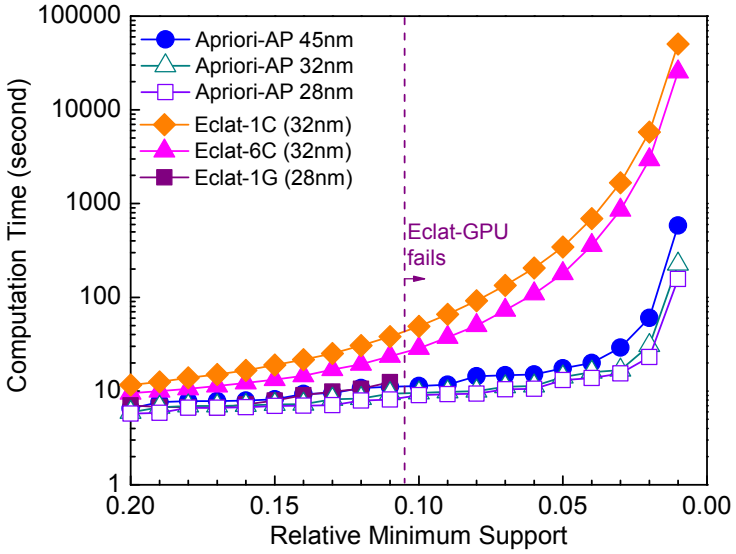
Performance Evaluation – vs. Eclat



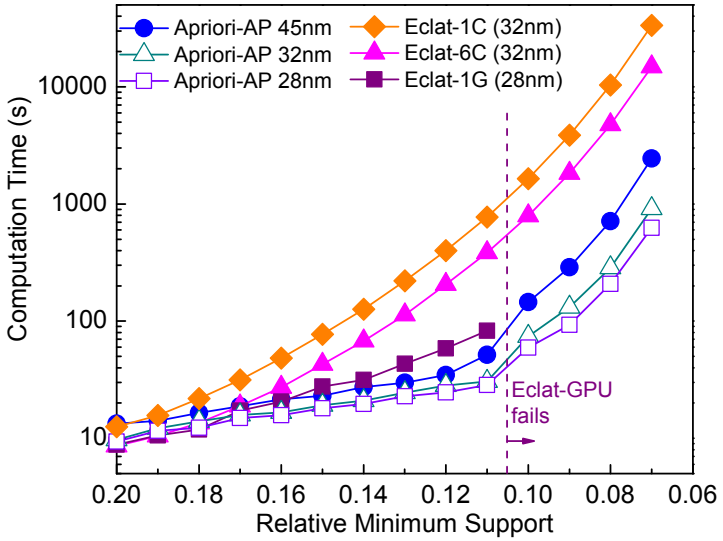
Accidents (34MB)



T100D20M (6.3GB)

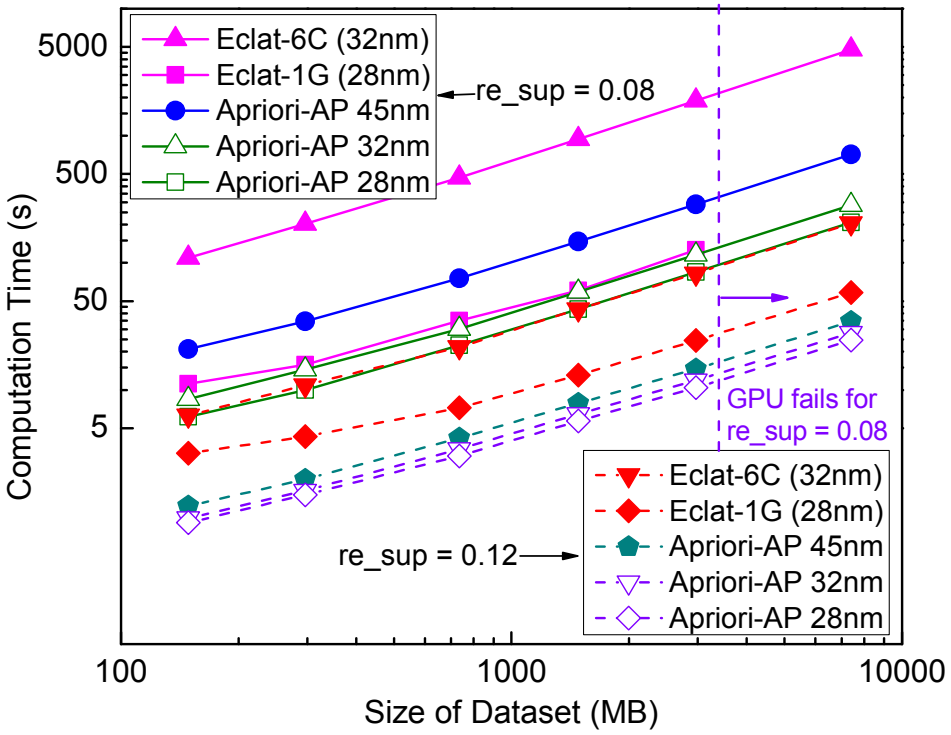


ENWiki(3.0GB)

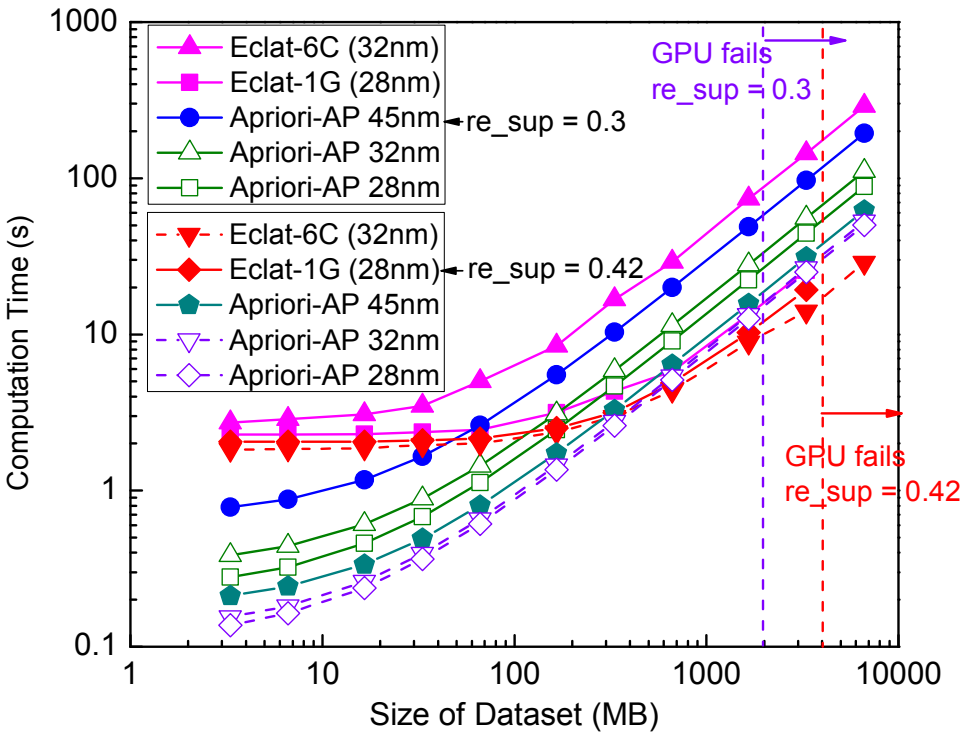


Webdocs5X (7.1GB)

Performance Evaluation – Data size



Webdocs



T100

Webdocs: the different data sizes are obtained by randomly sampling the transactions or by concatenating duplicates of the whole dataset of Webdocs

T100: the datasets with different sizes are obtained by varying the number of transactions using the IBM synthetic data generator

Conclusions and Future Work

- Hardware-accelerated ARM solution using Micron's AP architecture - MISD
- Novel automaton design for ARM
- Several optimization strategies:
  - IO minimization
  - Multiple-entry NFA to avoid routing reconfiguration
  - Concurrent mining
- Proposed AP design can match and count up to 18,432 itemsets in parallel on an AP D480 48-core board
- Up to 129X speedup over single-core CPU implementation of Apriori
- Outperforms the multicore-based and GPU-based implementations of Eclat ARM with up to 49X speedups, especially on large datasets
- Technology projections suggest even better speedups possible relative to the equivalent-process node of CPUs and GPUs
- Studies on data size demonstrate the memory constraint of parallel Eclat ARM, particularly for GPU implementation but nice scalability of our AP solution
- Will use the AP to accelerate other complex pattern mining tasks such as frequent sequential pattern mining and frequent episode mining for next step