Fast Approximation of the Top-k Items in Data Streams Using a Reconfigurable Accelerator

Ali Ebrahim and Jalal Khlaifat

Department of Computer Engineering

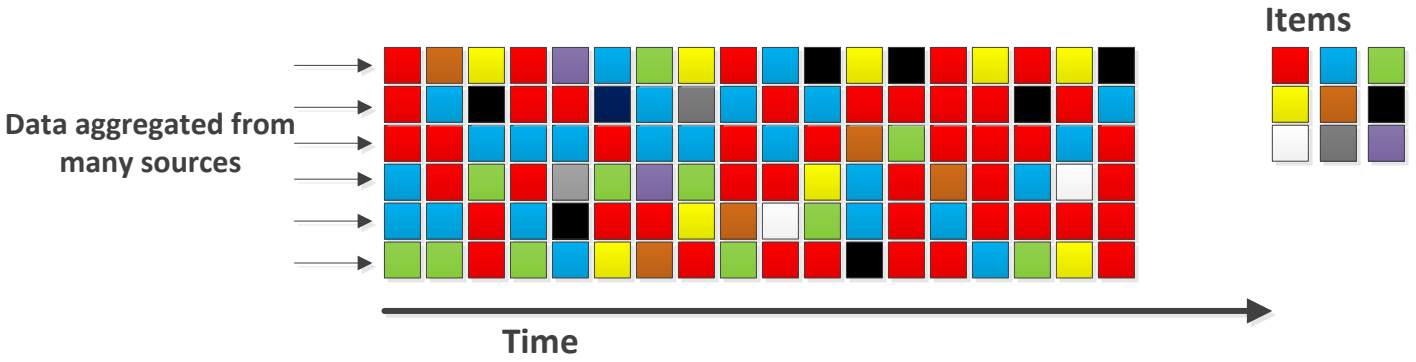
University of Bahrain, Bahrain

Outline

- **Introduction:**
  - Data Stream Model and Top-k Items Problem
- **Background:**
  - Previous Efforts
- **Proposal (Probabilistic Sampling):**
  - Architecture Overview
  - System Operation
  - FPGA Implementation
- **Results**
- **Summary**

The Data Stream Model

- Data is modelled as a "stream":
  - Large continuous feed of items
  - Items can be: visited websites, search queries, stocks, etc.

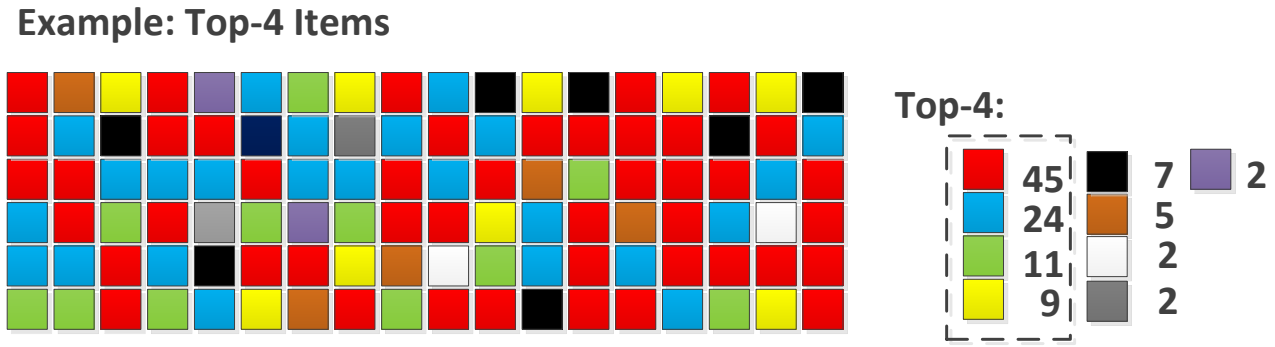


- ➢ Data is processed once to extract information
- ➢ One pass over input (no random access!)

Top-K Items in Data Streams

- Fundamental problem, research date back to 1982:

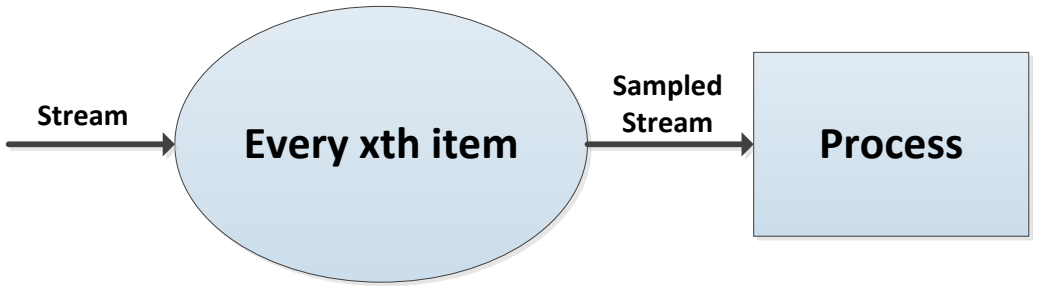
"Given a stream *S* of *N* items, what are the top-*k* items in *S*"



- Naïve solution: count every distinct item(impractical!)
- Approximation required!

Top-K Items in Data Streams

- Approximate algorithms:
  - Throughputs range: 10-30 Million Items/s [Cormode et al., 2008]
  - Sometimes, sampling is required to keep up with input throughput:



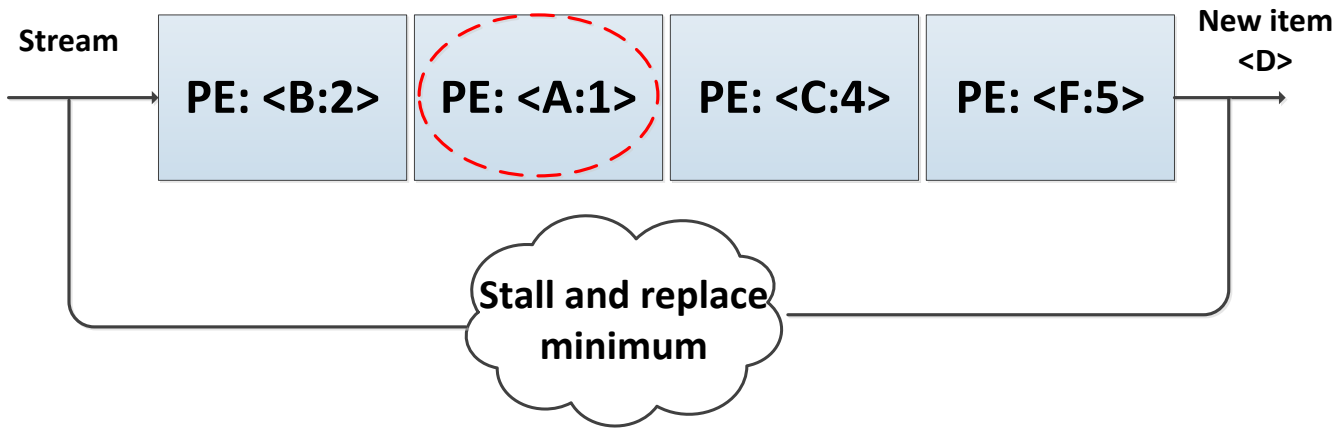
  - Low sampling rates affects accuracy!
  - Accelerated implementations desirable

Previous Efforts

- Space-Saving algorithm [Metwally et al., 2005]:
  - Use m item counters to monitor first m distinct items
  - Increment an item counter every time the item is re-encountered
  - A new item, replaces the item with the minimum count. Overestimate counts of new items
  - When stream is exhausted, take k items with highest counts
  - The larger m, the better the results!

Previous Efforts

- Systolic array architectures:
  - A Processing Element (PE) for each item

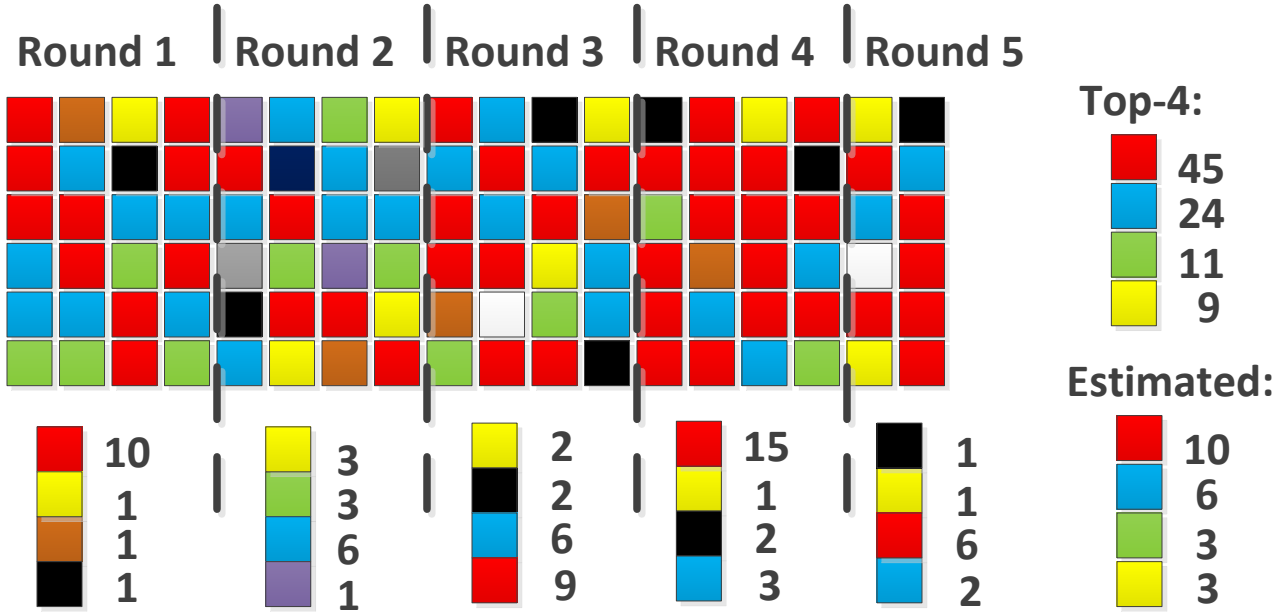


  - Needs a way to know position of minimum counts
  - A stall for every new item
  - Data distribution affects throughput

* A. Ebrahim and J. Khlaifat, "An Efficient Hardware Architecture for Finding Frequent Items in Data Streams," 2020 IEEE 38th International Conference on Computer Design (ICCD), 2020

Proposal: *Probabilistic* Sampling

- Probabilistic Algorithm [Demaine et al., 2002]
  - Divide stream into rounds of size r. Using m counters monitor first m distinct items, ignore all other items
  - Find top-k items in each round, and merge results



Probabilistic Algorithm

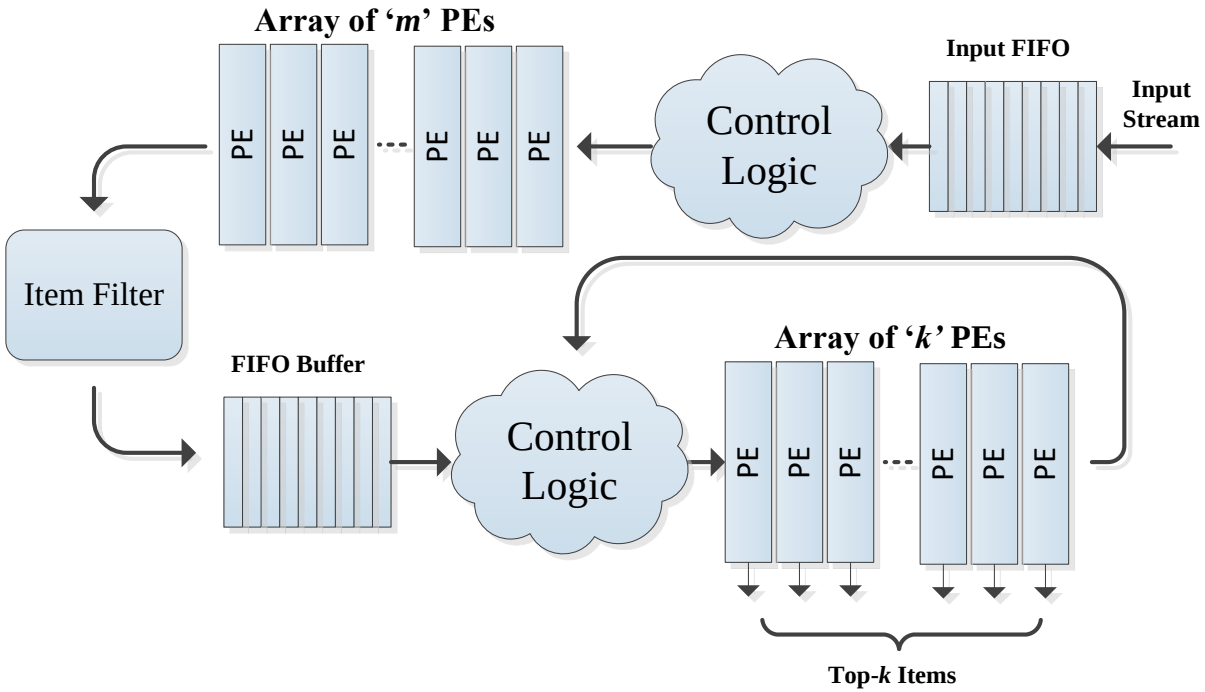
- For known N, $r = \sqrt{mN}$
- For unknown N, guess $N = 1,2,4, \ldots . 2^j$

Algorithm: Probabilistic

```
for each round of r items in S do
    for each item in round do
        if item is stored then
            Increment item's count
        else if stored items < m then
            Store item and set its count to 1
    Update a record of k distinct items with highest round counts
    Clear all m stored items and counts
```

Hardware Architecture

- Algorithm mapped to two systolic arrays
- One array counts the items in a round, the other keeps track of the top-k items

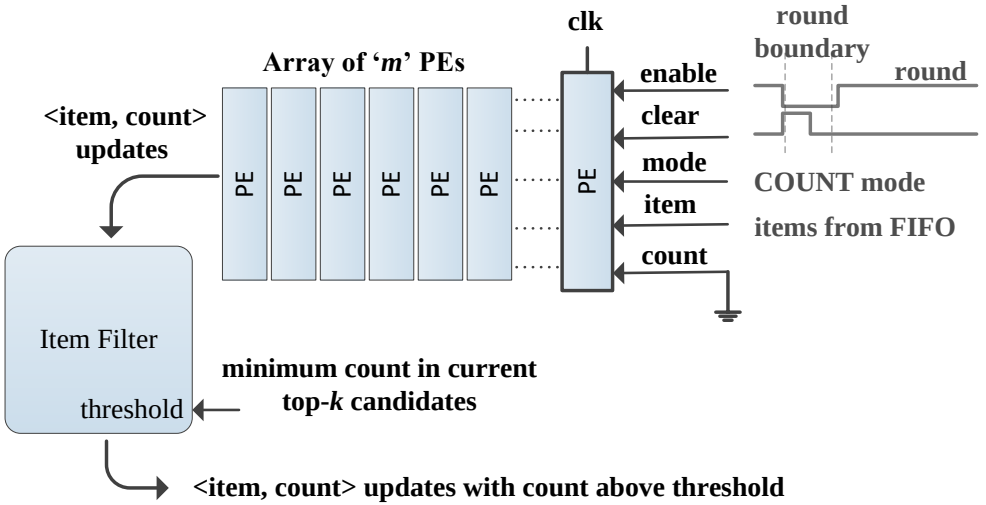


System Operation

Item Counting:

- Count first *m* items within a round
- Forward a count update for each new hit
- Filter updates bellow threshold
- Reset PEs between rounds

- **Non-stalling array**
- **Less complex PEs**
- **Smaller count registers needed**



Algorithm: COUNT Mode

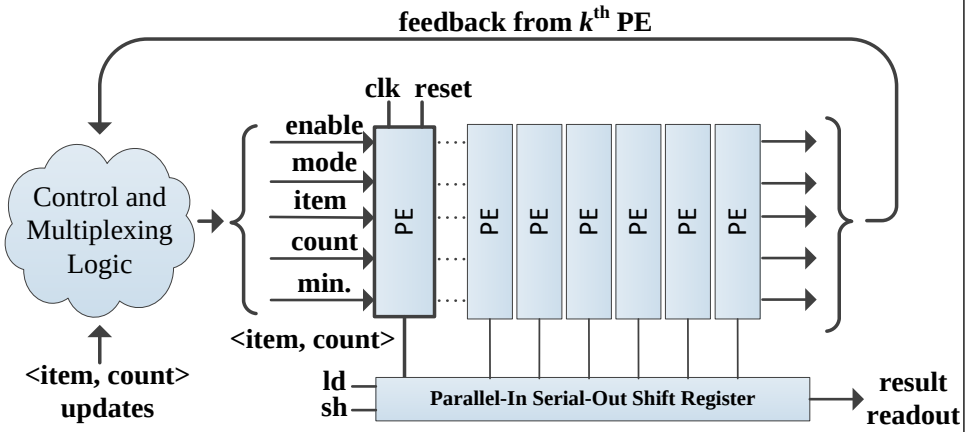
```
// Upon receiving a new item in COUNT mode
if PE is empty then
    Store item and set count to 1
    Pass <item, count> pair to next PE in FORWARD mode
else if incoming item = stored item then
    Increment count
    Pass <item, count> pair to next PE in FORWARD mode
else
    Pass incoming item to next PE in COUNT mode
```

System Operation

Track top-k items:

- PEs monitor top-k candidates
- UPDATE Mode:
  - Update counts of current top-k candidate
  - Shift position of the PE with minimum count out of the array

➢ **A new candidate needs to replace item with minimum count**



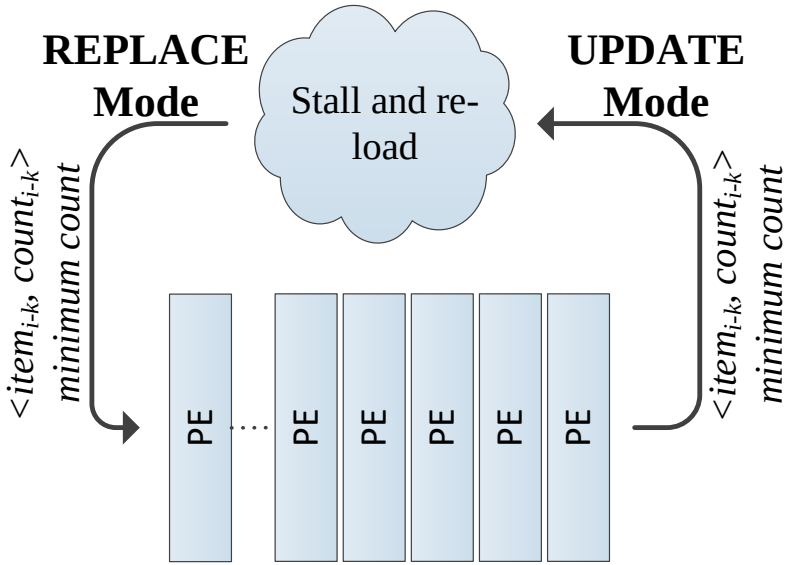
Algorithm: UPDATE Mode

```
// Upon receiving an <item, count> pair and min. count
   value in UPDATE mode
if PE is empty then
    Store <item, count> pair
else if incoming item = stored item then
    if incoming count > stored count then
        Update stored count
else
    Pass incoming <item, count> pair to next PE in UPDATE mode
    Pass min(stored count, incoming min. count value) to next PE
```

System Operation

Track top-k items:

- REPLACE Mode:
  - Stall and reload
  - Doesn't happen frequently!
  - Congestions on buffer eased by filtered items



➢ **Better than Space-Saving!**

Algorithm: REPLACE Mode

```
// Upon receiving an <item, count> pair and min. count
   value in REPLACE mode
if incoming item = stored item then
    if incoming count > stored count then
        Update stored count
else if incoming min. count value = stored count then
    Replace current <item, count> pair
else
    Pass incoming <item, count> pair to next PE in REPLACE mode
    Pass incoming min. count value to next PE
```

FPGA Implementation

- Intel Arria 10 FPGA (10AX115N2F45E1SG)
- 32-bit registers, 256 FIFO depth

Array Size (*m*)	ALMs	Registers	f_{max} (MHz)
1 PE	57 (<1%)	132	606
1000	59223 (14%)	138021	359
2000	118231 (28%)	271037	348
3000	177185 (41%)	410128	346

Array Size (*k*)	ALMs	Registers	f_{max} (MHz)
1 PE	127 (<1%)	227	414
100	13180 (3%)	27218	343
200	26511 (6%)	54820	329
300	39823 (9%)	79284	306

FPGA Implementation

- 3000+ PEs, 200+ MHz operating frequency
- More than double the PEs compared to Space-Saving

Accelerator Size	ALMs	Registers	Block RAM	f_{max} (MHz)
k = 100, m = 3000	184056 (43%)	430105	24576 (<1%)	308
k = 200, m = 3000	200278 (47%)	456424	24576 (<1%)	290
k = 300, m = 3000	216450 (51%)	484125	24576 (<1%)	276

Results: Throughput

- Test with synthetic datasets (Zipfian distribution, α from 1.0 to 2.5)
- Throughput independent of data skew (1 update/clk cycle)
- Speedup : **20x** compared to software
  1.8x compared to FPGA (Space-Saving)
  1.5x compared to GPU

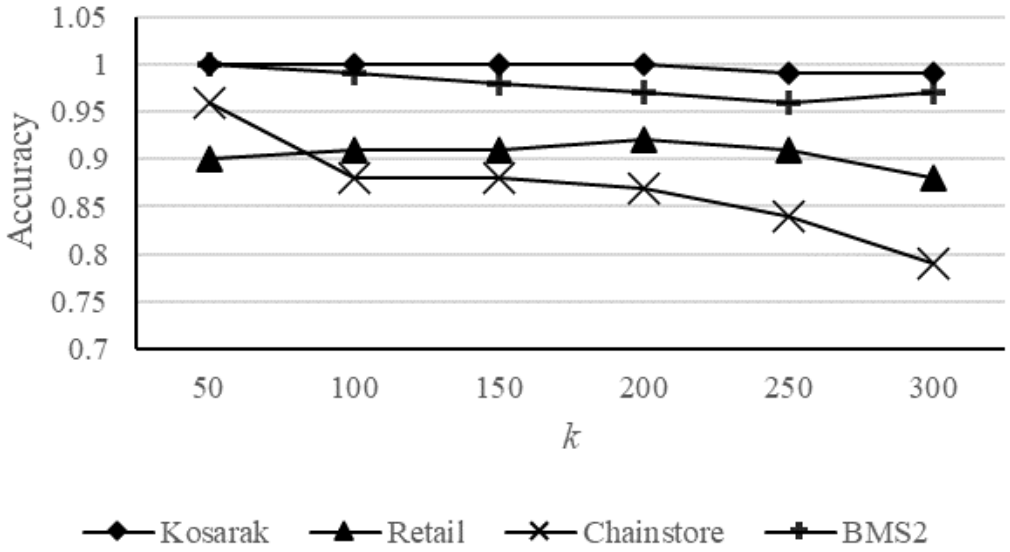
Implementation	Algorithm	Throughput (Million item/s)		
		Min.	Max.	Average
Software [2]	SSL	5	22	15
Xilinx Virtex-6 FPGA (1024 PEs) [11]	Space-Saving	95	95	95
Xilinx Virtex-6 FPGA (1024 PEs) [13]	Space-Saving	73	242	152
Intel Arria 10 FPGA (1200 PEs) [14]	Space-Saving	101	200	174
GPU [21]	GSS-Ballot	12	27	18
GPU [22]	GPUSB	207	207	207
Proposed (Top-100)	*Probabilistic*	308	308	308
Proposed (Top-200)	*Probabilistic*	290	290	290
Proposed (Top-300)	*Probabilistic*	279	279	279

Results: Accuracy

- Real datasets:

Dataset	Distinct Items	Size
Retail	16469	908399
Kosarak	41270	8019015
Chainstore	46086	8042879
BMS2	3340	358278

- Test for top-100 to top-300



- **Average: 90%**
- **Worst case: 74%**

Summary

- Significant throughput gains possible with FPGAs for accelerating the top-k items problem
- Probabilistic algorithm seems like a good fit for FPGAs
- There is room for further improvements
- Future work:
  - Increase scalability and accuracy
  - Maintain throughput

Thank You!