# Data stream statistics over sliding windows: How to summarize 150 Million updates per second on a single node

Grigorios Chrysos†, Odysseas Papapetrou‡, *Dionisios Pnevmatikatos*†, Apostolos Dollas†, Minos Garofalakis†∗

†Technical University of Crete, Greece

‡Eindhoven University of Technology, Netherlands

∗ATHENA Research and Innovation Center, Greece

# Why process data streams in real-time?

Real time, continuous, high-volume data streams:

- Network monitoring for DoS attacks
- Monitoring market data to guide algorithmic trading
- Adaptive online advertising, etc.

Too big to store in memory => build approximate *sketch synopses*

Our focus here: Exponential Count-Min (ECM) sketches

- Papapetrou *et al* [VLDB12, VLDBJ15]
- Space and time efficient
- Support *frequency* and *inner product* queries
- Bounded error data structures

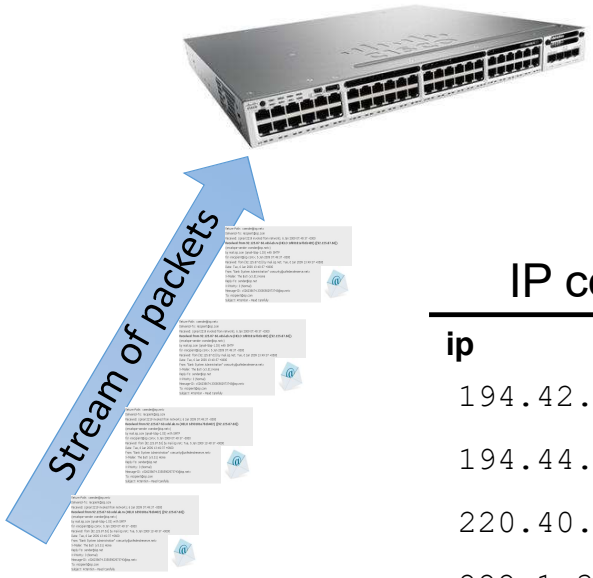
**Contribution: Explore ECM Sketch acceleration architectures on FPGA**

# Outline

- ECK Sketch Primer
- ECM Acceleration Architectures
- Evaluation
- Conclusions

# Example: Distribution statistics at routers

- Maintain *sliding-window data stream statistics*




IP counters

| ip | freq. |
|---|---|
| 194.42.1.1 | 2 |
| 194.44.2.6 | 0 |
| 220.40.41.4 | 1 |
| 222.1.34.7 | 1 |

1000 msec sliding window

| IP address | Timestamp (msec) |
|---|---|
| 194.42.1.1 | 0 |
| 194.44.2.6 | 2 |
| 194.42.1.1 | 4 |
| 220.40.41.4 | 7 |
| 194.42.1.1 | 8 |
| … | |
| 220.40.41.4 | 999 |
| 222.1.34.7 | 1001 |
| 194.42.1.1 | 1003 |
| 194.42.1.1 | 1009 |

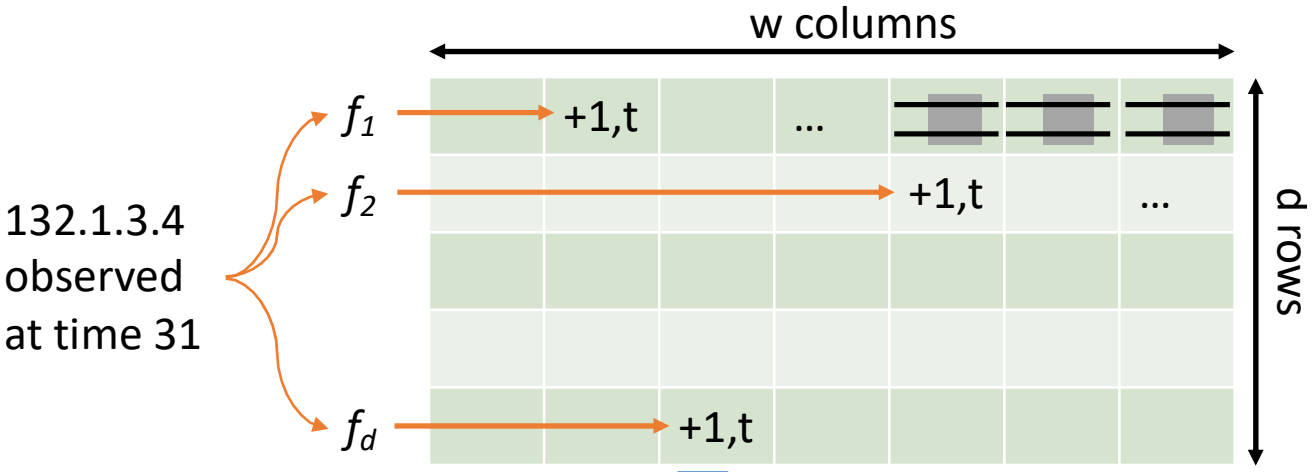
# ECM Sketch Primer

- Sketch is a set of *d* hash functions f1, f2, . . . , fd and a 2-dimensional array of *w x d* “counters”

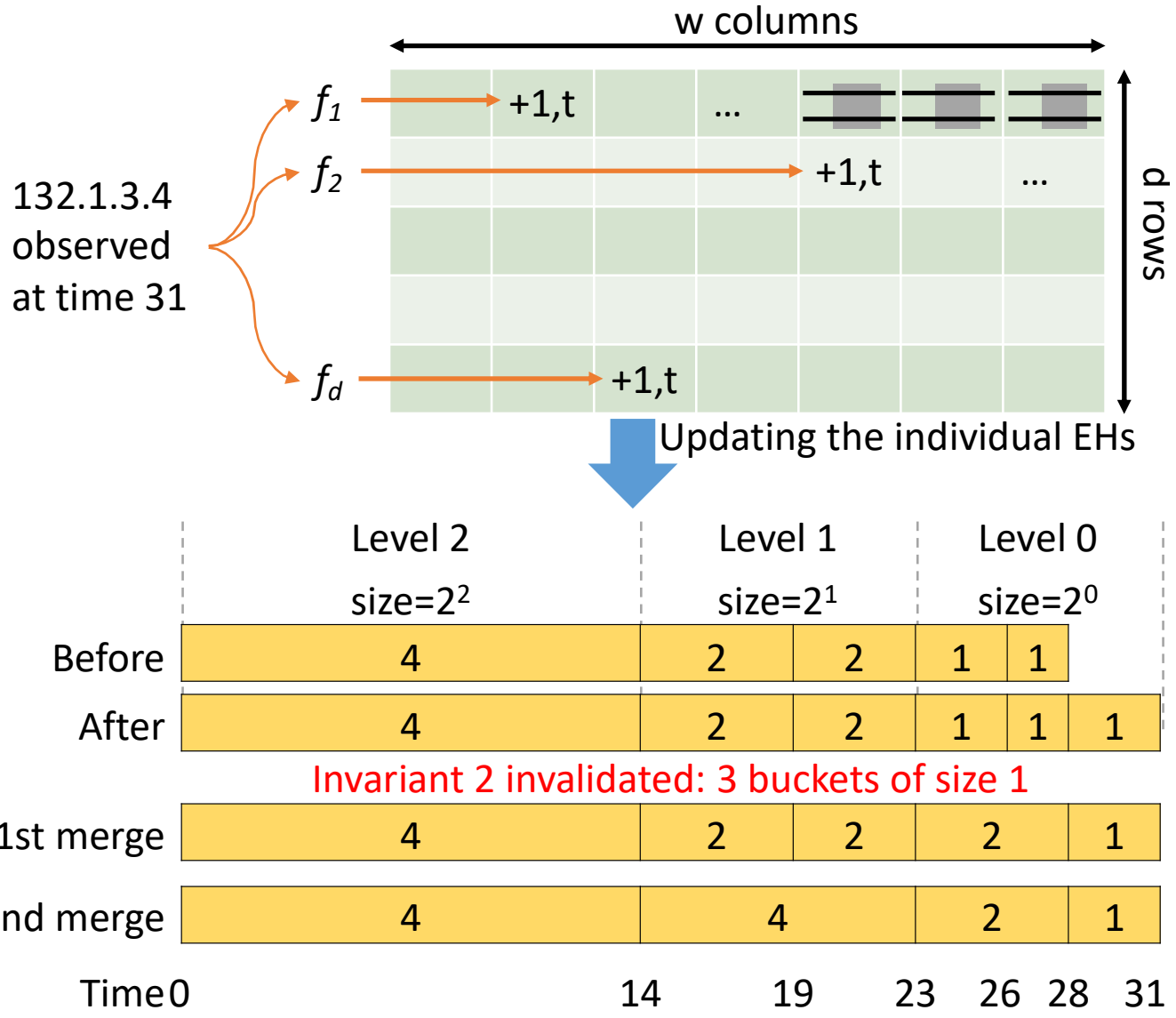
- “Counter” is an Exponential Histogram structure (space efficient for large time windows)

- For each incoming key:
  - Is hashed *d* times to select which EH to update in each of the *d* rows
  - d EHs are updated

# ECM Sketch



# ECM Sketch


w columns
d rows
132.1.3.4
observed
at time 31
$f_1$
$f_2$
$f_d$
+1,t
+1,t
+1,t
...
...
Updating the individual EHs
Level 2
size=$2^2$
Level 1
size=$2^1$
Level 0
size=$2^0$
Before
4
2
2
1
1
After
4
2
2
1
1
1
Invariant 2 invalidated: 3 buckets of size 1
1st merge
4
2
2
2
1
2nd merge
4
4
2
1
Time 0
14
19
23
26
28
31

# Sizing ECM Sketches

ECM sketch provides frequency estimates with an error less than ε*N, with probability at least 1 – δ
N denotes the length of the sliding window

**ECM Sketch parameters:**

- Number of rows: d = ⌈ln 1/δ⌉
- Number of Exponential Histograms (EHs) in each line : w = ⌈e/ε⌉
- Number of positions at each bucket level: k = ⌈1/ε⌉
- Number of bucket levels for each EH: L >= O(log(2N/k) + 1)

**Update complexity:** O(logN)

**Amortized complexity** is **constant**, expected 2 merges per update

# Outline

- ECK sketch primer
- ECM Acceleration Architectures
- Evaluation
- Conclusions

# Accelerator Architecture #1

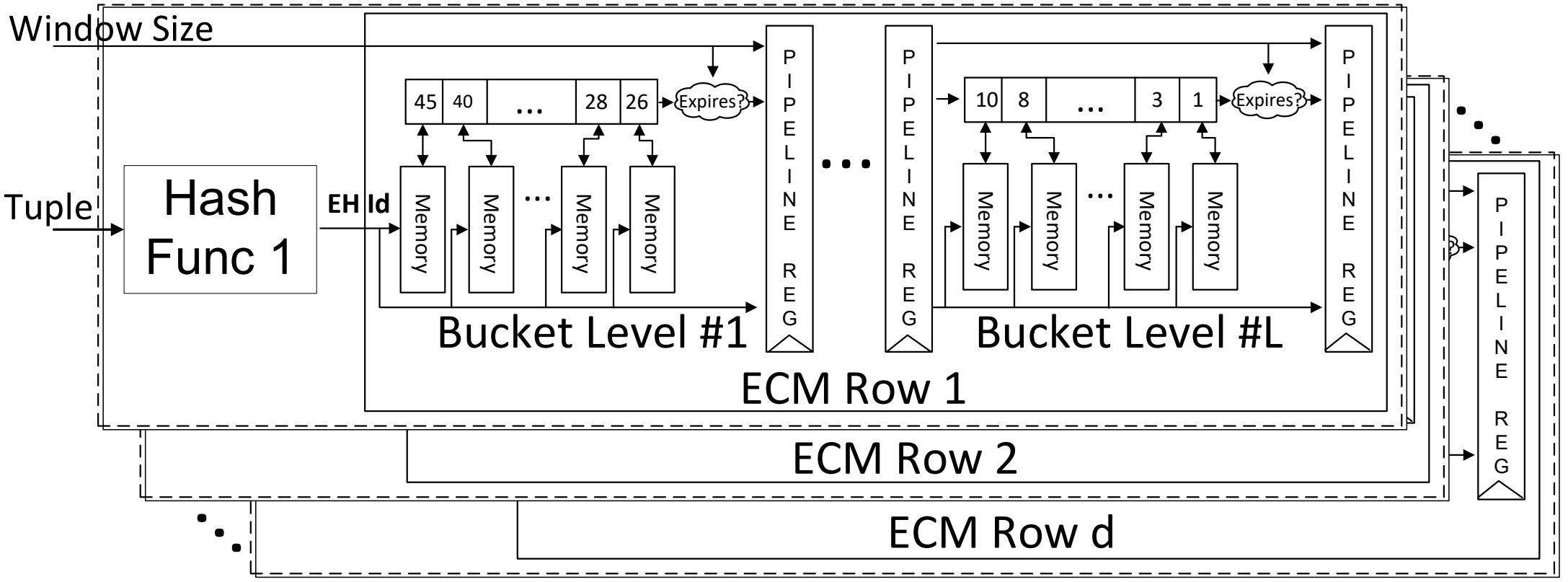
ECM Sketches are 3-D structures *d x w x L*

- Only *one* EH per row active at any time
- Have *d* independent structures
- Group data for each of the *w* EHs of a ECM row in BRAMs
- Update takes >=1 cycle, but pipelined!

Result:

+ Fully pipelined, guaranteed throughput design
- Worst case design: each EH has L pipeline stages, only 2 active on the average

# Fully pipelined architecture (FC)




**Problem**: Did not fit in Convey V6 FPGA due to high BRAM use

# Accelerator Architecture #2

Our Convey HC-2ex platform uses Virtex6 devices

=> Not particularly large devices

Together with “shell”, the FC architecture did not fit

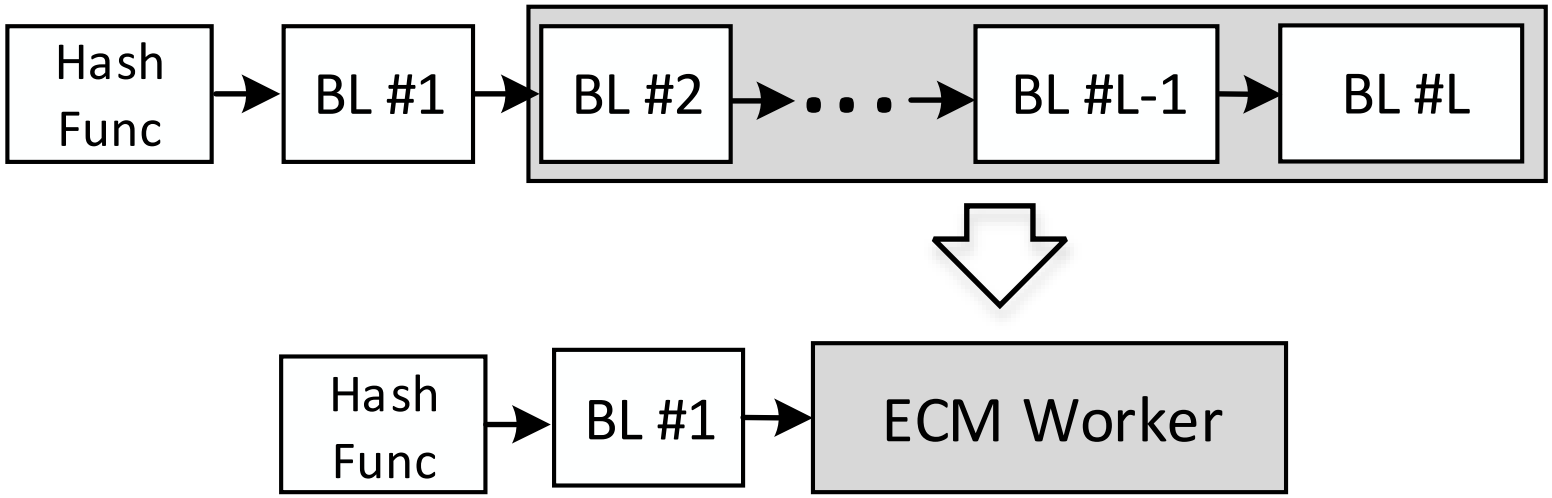
BRAM space was the bottleneck

Go for space efficiency:

BRAMs underused (w is 55, minimum BRAM rows is 512)

Amortized update cost is 2 => most pipelined levels are idle!

# Key idea to exploit amortized ECM update cost



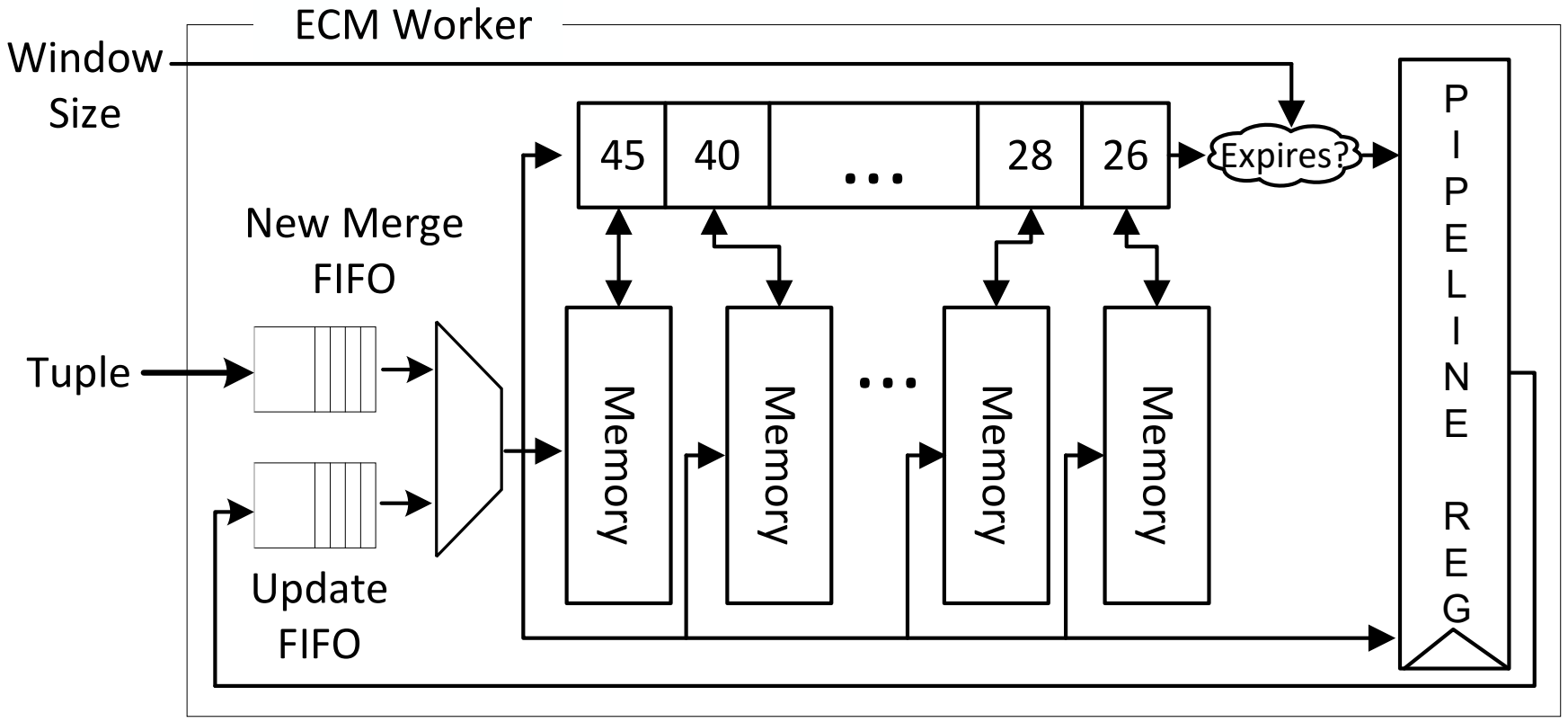

**CAUTION**:

Space: mapped L-1 level counters into one worker (BRAM size?)

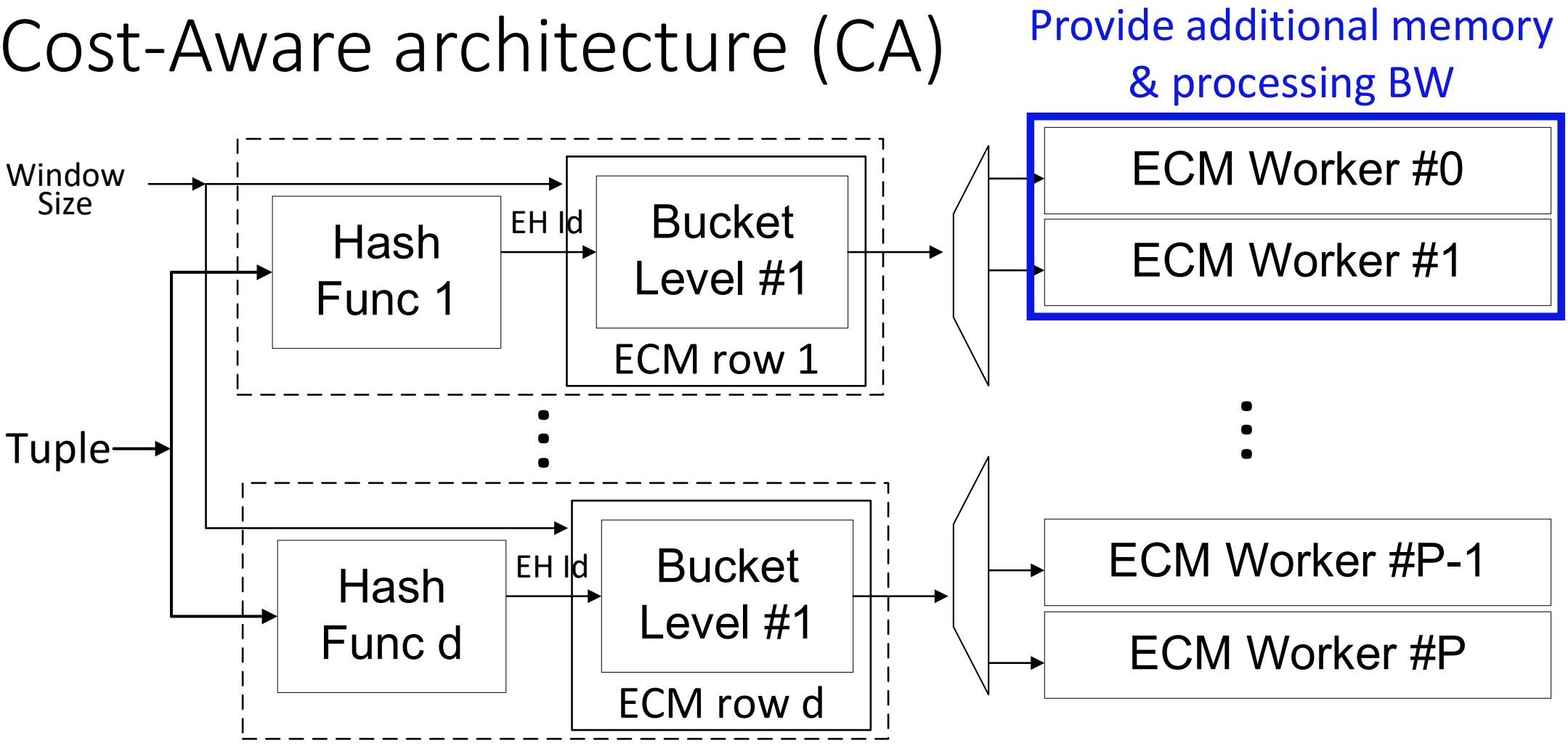
Multiple hits in the same row => more work for worker

Multiple hits in the same EH => more work for worker

# ECM Worker Internal Structure



# Cost-Aware architecture (CA)



# Now we can play:

One parameter: how many bucket levels to instantiate *before* Worker

- More levels => better tolerance to skewed workloads

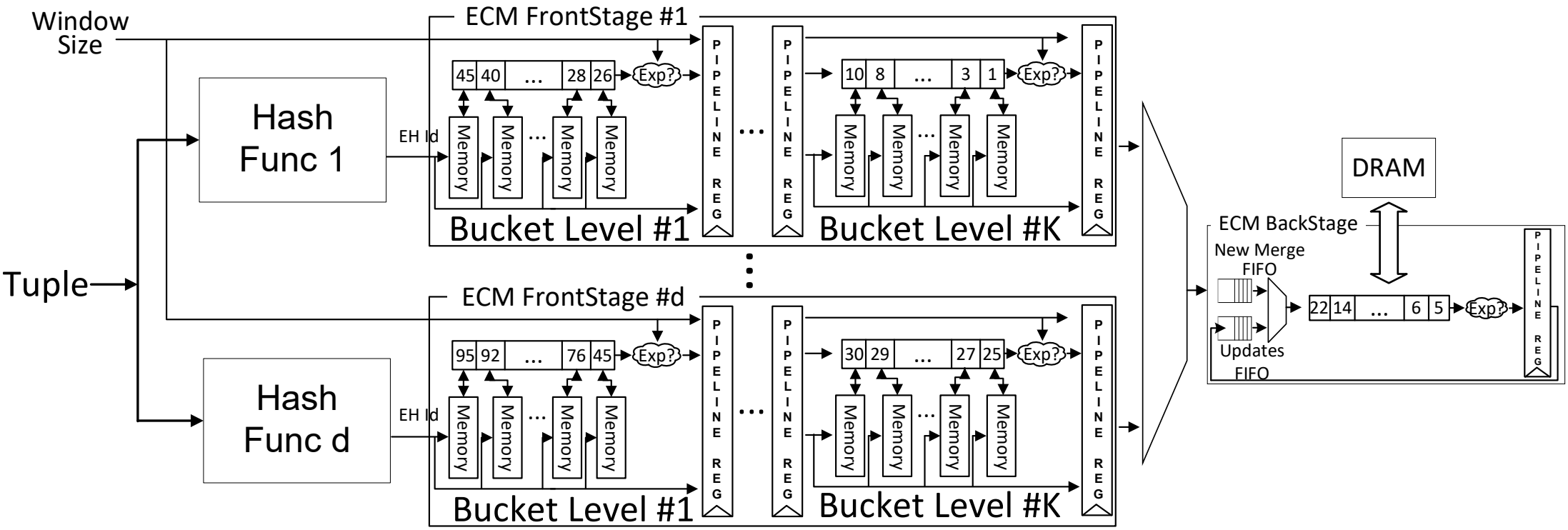
What about LARGE windows?

L becomes large BUT update load exponentially decreases

=> store in DRAM!

DRAM is slower than BRAM => need to get there infrequently

# Hybrid Architecture




**CAUTION**:

Choose K carefully so that DRAM BW is sufficient *most* of the time

# Can we Exceed 1 tuple per cycle?

All architectures so far assume input of one tuple per cycle

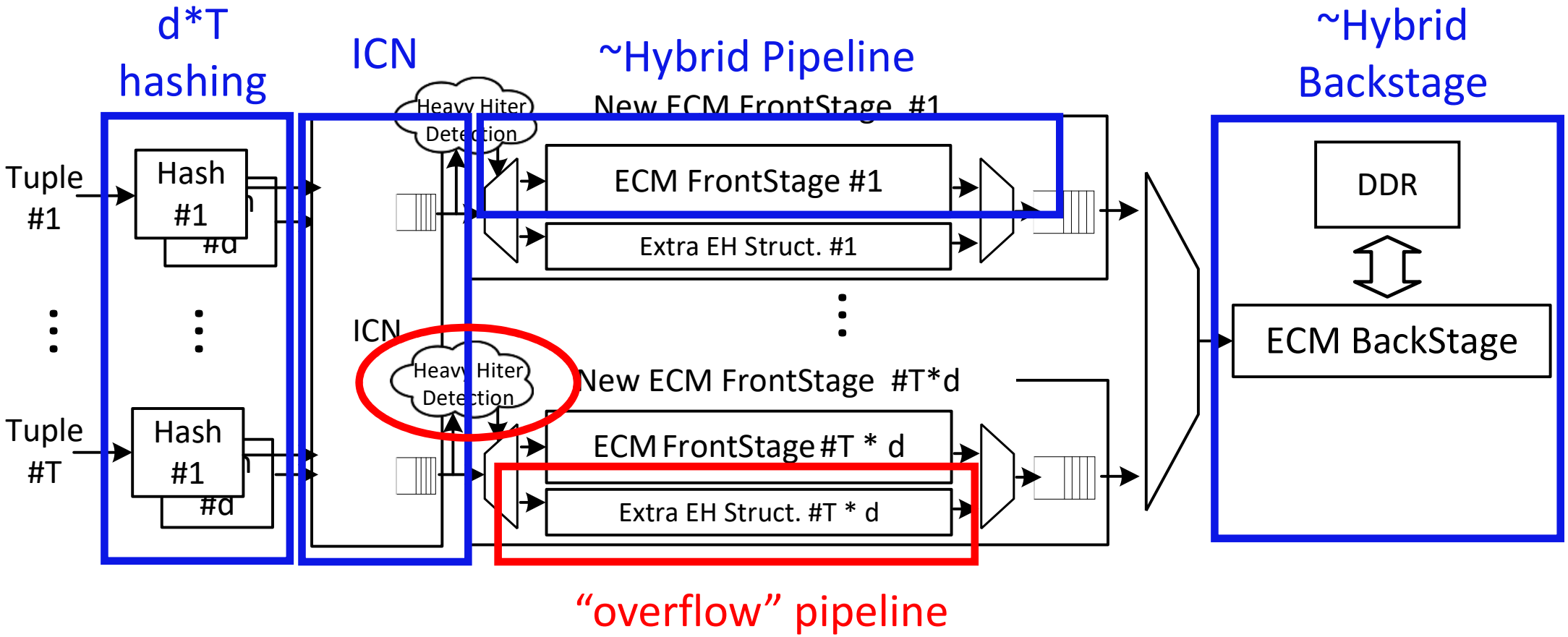
What if I have T input tuples per cycle?

- Hash d*T tuples
- Update d*T EHs
- If d*T << #EHs, chances are good that different EHs will be updated

Corollaries:

- Cannot group into d rows (d << d*T)
- Multiple updates to same EH at same cycle are possible!

# > 1 tuple per cycle: Multithreaded Architecture



# Outline

- ECK sketch primer
- ECM Acceleration Architectures
- Evaluation
- Conclusions

# System implementation

## System Parameters

$\varepsilon = 0.05$, $\delta = 0.05$

$w = 55$, $d = 3$, $k = 11$

CA architecture P was set to 6 (2 workers per row)

Hybrid: K = 5 (bucket levels before DRAM)

MT: K = 5, T = 3, #FrontStages = 10

## Target platform

Convey HC-2ex, two six-core Xeon E5-2640 processors, 128GB and four Xilinx Virtex-6 LX760 FPGAs (use *only one*)

- Shell logic clock fixed at 150MHz
- 474K LUTs, 948K flip flops, and 1440x18 Kbit BRAMs

# Evaluation

## Five Input Datasets

- ✓ Crawdad SNMP Fall 03/04 [11]
- ✓ CAIDA Anonymized Internet Traces 2011
- ✓ WC, the data set from world cup98 [2]
- ✓ Two randomly generated traces

## Software baseline

- Reference software from Papapetrou et al. [VLDBJ15]
- Multi-thread parallelized version of the reference SW (lock limited)

## FPGA versions

- Implemented & tested on Convey

# Performance comparison (single FPGA)

| Dataset | #Tuples | Update Rate (Million ($10^6$) Tuples/sec) | | | | |
|---|---|---|---|---|---|---|
| | | SWx1/x24 | FP | CA | Hybrid | MT |
| Random1 | $10^8$ | 10.6/16.4 | *150*† | 145.1 | 101.3 | 178.2 |
| Random2 | $10^8$ | 10.8/19.9 | *150*† | 147.3 | 101.2 | 177.8 |
| SNMP | $3.1 * 10^7$ | 11.4/26.6 | *150*† | 141.1 | 101.3 | 173.0 |
| CAIDA | $10^8$ | 10.2/19.6 | *150*† | 147.9 | 101.2 | 183.3 |
| WC | $10^8$ | 12.2/24.6 | *150*† | 147.1 | 101.1 | 148.5 |

**Note:**

SW performance is between 10-27 Mtuples/sec

† FP operating frequency is estimated

FP performance is guaranteed, {CA, Hybrid, MT} are best effort

# Resource utilization

| Virtex6 Resources | FP | CA | Hybrid | MT |
|---|---|---|---|---|
| LUTs | 137,9K/29% | 22,3K/5% | 86,3K/18% | 223,3K/47% |
| FFs | 57,0K/6% | 5,7K/1% | 38,5K/4% | 141,6K/15% |
| BRAMs | 1071/74% | 357/25% | 651/45% | 847/59% |

Numbers *DO NOT* include the “shell” logic

CA is more cost effective than FP (6x in logic, 3x in BRAMs)
MT cost is significant, Hybrid is affordable

FP & CA are the best overall options

# Performance on Recent Devices: US+ xczu17eg

| UltraScale Resources | FP | CA | Hybrid | MT |
|---|---|---|---|---|
| LUT | 62.6K/15% | 26.1K/6% | 35.8K/8.5% | 371.6K/87% |
| FF | 21.5K/2% | 8.7K/1% | 6.8K/1% | 110.4K/13% |
| BRAM | 535/67% | 220/28% | 168/21% | 504/63% |
| Freq (MHz) | 260 | 222 | 244 | 170 |
| Performance (Mtuples/sec) | 260 | 214 | 165 | 198 |

**Note:**

Post P&R tool result

FP is affordable,  CA is even better (in cost)!

Hybrid and MT are not really worth it

# Conclusions

Sliding-window statistics on streaming data is an important application domain

ECM Sketches offer error bound in common queries and are HW friendly

A range of efficient accelerators is possible and offer 5-10x compared to multithreaded SW

Guaranteed or best-effort operation? Cost vs Error tolerance tradeoff!

Additional resources in modern FPGAs can be used to implement better ECM sketches: larger time window and/or tighter error bounds $\varepsilon$ and $\delta$

# Thank you!

# Questions?

This work was supported in part by EU projects:

- FP7 Qualimaster (#619525)
- FET-HPC EXTRA (#671653)
- Marie Sklodowska-Curie MSCA-COFUND-2017 project AQuViDa (#665667)