Timely Detection of Heavy Hitters in External Memory

Michael A. Bender
Stony Brook University

J. W. Berry
Sandia National Laboratories

Martin Farach-Colton
Rutgers University

Rob Johnson
VMware Research

Thomas Kroeger
Sandia National Laboratories

Prashant Pandey
Carnegie Mellon University

Cynthia Phillips
Sandia National Laboratories

Shikha Singh
Williams College
The Heavy Hitter Problem

- Also called the **frequent items problem**
- Stream of $N$ elements arrive over time
- A heavy hitter is an element that occurs at least $\phi N$ times
- Usually reported at the end of the stream
- **Hard in small space**: exact solution requires $\Omega(N)$ words

[Cormode 05]
Timely Heavy Hitters: Online Event Detection Problem (OEDP)

- Stream of elements arrive over time
- An **event** occurs at time $t$ if $s_t$ occurs exactly $T = \phi N$ times in $(s_1, s_2, \ldots, s_t)$
- In the **online event detection problem (OEDP)**, we want to report all events **as soon as** they occur.

Suppose $T = 4$
OEDP Requirements

• Stream is **large & high-speed** (millions/sec)

HIGH THROUGHPUT (fast inserts)
OEDP Requirements

• Stream is large & high-speed (millions/sec)

HIGH THROUGHPUT (fast inserts)

• Events are high-consequence real-life events

Immediate reporting (ONLINE)

NO ERRORS (esp. false negatives)

Every insert is also a query!
OEDP Requirements

- Stream is **large & high-speed** (millions/sec)
  - **HIGH THROUGHPUT** (fast inserts)

- Events are **high-consequence real-life events**
  - **Immediate reporting (ONLINE)**

- Very small reporting threshold $T << N$ (stream size)
  - **Be scalable to SMALL THRESHOLDS**

Every insert is also a query!
Firehose Streaming Benchmark

• Department of Defense (DoD) and Sandia designed the Firehose benchmark for this setting [https://firehose.sandia.gov/]

• The high-speed input stream consists of (key, value) pairs

• On the 24th occurrence of key appears, some function of its values must be reported immediately

• Most difficult part of this is determining when the 24th instance of a key arrives

\[ T = 24 = o(1) \]

Stream size ~ 1 TB
One-Pass Streaming Has Errors

- Exact one pass solution requires $\Omega(N)$ space
- Approximate solutions trade off accuracy for space [Alon et al. 96, Berinde et al. 10, Bhattacharyya et al. 16, Bose et al. 03, Braverman et al. 16, Charikar et al. 02, 05, Demaine et al. 02, Dimitropoulos et al. 08, Larsen et al. 16, Manku et al. 02, Misra and Gries 82, etc.]

Stream

Maintain count estimates in RAM [Misra & Greis 82]
Two-Pass Streaming Isn’t Real Time

- A second pass over the stream can get rid of errors

No errors but offline!

Maintain count estimates in RAM [MG82]
Exact Solutions Need Large Space

- A second pass over the stream can get rid of errors
- To do a second pass, you need to store your data somewhere

No errors but offline!

Maintain count estimates in RAM [MG82]
If Data is Stored: Why Not Access It?

- A second pass over the stream can get rid of errors
- To do a second pass, you need to store your data somewhere
- Why wait for second pass?

If Data is Stored: Why Not Access It?

- A second pass over the stream can get rid of errors
- To do a second pass, you need to store your data somewhere
- Why wait for second pass?

External Storage

- Maintain count estimates in RAM [MG82]

No errors but offline!
Modern External Memory: SSDs

Sequential access on modern SSDs ~ Random access in RAM!

Random accesses are slow, but fine if not bottleneck
The External-Memory Model

- Data is transferred between RAM and EM in blocks of size $B$
- Performances measured in # of I/Os
External-Memory Model: Review

- **Question:** How many I/Os to scan an array of length $N$?

- **Answer:** $O(N/B)$ I/Os.
External-Memory Model: Review

• **Question:** How many I/Os for a point query or insert into a B-tree with N elements?

• **Answer:** $O(\log_B N)$
**Optimal Trade-Off Curve** [Brodal, Fagerberg 03]

- Logging: inserts are fast, but queries are slow
- B-trees: point queries are fast but inserts are slow
Idea Behind Write-Optimization

- You can improve insert costs without losing out on queries
Does Write-Optimization Solve OEDP?

• Write optimized data structures like COLA, cascade filters, etc. (WODs) let you do fast inserts and B-tree like queries

<table>
<thead>
<tr>
<th>Insert</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O\left(\frac{\log N/M}{B}\right)$</td>
<td>$\Omega(\log_2 N)$</td>
</tr>
</tbody>
</table>

Example: WODS dictionaries like cascade filters/ COLA do not solve the problem! But we can use insights from WODs
What We Do

- Combine streaming and WODs techniques to solve OEDP: design cache-efficient variant of the classic HH algorithm

- Can achieve **immediate reporting with no errors** at a cost that slightly worse than best insertion cost

<table>
<thead>
<tr>
<th>Optimal Insert</th>
<th>Optimal Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O\left(\frac{\log N/M}{B}\right))</td>
<td>(\Omega(\log_B N))</td>
</tr>
</tbody>
</table>

- If **slight delay in reporting** is allowed, we present a new data structure that matches the optimal insertion cost
Our Results

- Given a stream of size $N$ and $\phi N > \Omega(N/M)$, the amortized cost of solving OEDP is

$$O\left(\left(\frac{1}{B} + \frac{1}{(\phi - 1/M)N}\right)\log \frac{N}{M}\right)$$

If $\phi N > B$ or $N > MB$, this reduces to

$$O\left(\frac{\log N/M}{B}\right)$$
Our Results

• Given a stream of size $N$ and $\phi N > \Omega(N/M)$, the amortized cost of solving OEDP is

$$O\left(\frac{1}{B} + \frac{1}{(\phi - 1/M)N} \log \frac{N}{M}\right)$$

If $\phi N > B$ or $N > MB$, this reduces to

$$O\left(\frac{\log N/M}{B}\right)$$

• Allowing a constant time stretch in reporting, we can support arbitrarily small thresholds $\phi$ with amortized cost

$$O\left(\frac{\log N/M}{B}\right)$$
Approximate Heavy Hitters Problem

- All items with count at least $\phi N$ must be reported.
- No item with count $< (\phi - \epsilon)N$ should be reported.
- Items count in between may or may not be reported (if reported these items are false positives).

For exact, set $\epsilon = 1/N$.
Misra Gries (MG) Algorithm

- Generalization of Moore 81 majority finding algorithm
- First proposed in 1982 by Misra and Gries, rediscovered twice in 2002, many improvements followed
- Finds $k$ items that appear at least $N/k$ times
- For AHH, set $k = 1/\varepsilon$

[J.Alg 2, P208-209] Suppose we have a list of $n$ numbers, representing the “votes” of $n$ processors on the result of some computation. We wish to decide if there is a majority vote and what the vote is.
Misra Gries (MG) Algorithm

- Maintain $1/\epsilon$ counters in memory
- When an item arrives
  - if there is a counter for it, increment the counter
  - if there is no counter for it
    - and there is space, add a counter and set to 1
    - otherwise, decrement all counters

[Cormode 05]
Misra Gries (MG) Algorithm

Counters

Items identified by color; # is the count

$\lceil \frac{1}{\varepsilon} \rceil$
Misra Gries (MG) Algorithm

\[ \left\lceil \frac{1}{\varepsilon} \right\rceil \]
Misra Gries (MG) Algorithm

\[ \lceil \frac{1}{\varepsilon} \rceil \]

1 2 3 4 2 1 2 ... 2 3 1

[1/\varepsilon]
Misra Gries (MG) Algorithm

\[ \lceil \frac{1}{\varepsilon} \rceil \]

\[ \begin{array}{cccccccc}
3 & 4 & 3 & 1 & 2 & 2 & 3 & 1 \\
1 & \ldots & & & & & & \end{array} \]
Misra Gries (MG) Algorithm

No counter for this item; No space to insert

1 3 4 3 1 2 ... 2 3 1 1

$\lceil 1/\varepsilon \rceil$
Decrement all counters

Misra Gries (MG) Algorithm

\[ \left\lceil \frac{1}{\varepsilon} \right\rceil \]
Misra Gries (MG) Algorithm

Remove if zero

$\left\lceil \frac{1}{\epsilon} \right\rceil$
MG Algorithm Analysis

• Let $\tilde{f}$ be the count estimate for item with frequency given by MG algorithm $f$

• Then

$\tilde{f} \leq f \leq \tilde{f} + N\varepsilon$

An item’s counter is incremented only when an instance of it is seen
MG Algorithm Analysis

- Let \( \tilde{f} \) be the count estimate for item with frequency given by MG algorithm \( f \)
- Then
  \[
  \tilde{f} \leq f \leq \tilde{f} + N\varepsilon
  \]

How many times can we lose a count of an item? Every time we lose an item count, we decrement all counters by one. Can happen only \( N/(1/\varepsilon) \) times!
MG for Approximate Heavy Hitters

- Run the MG algorithm and report all items with count estimate \( > (\phi - \varepsilon)N \)

- Satisfies AHH guarantees
  - If \( f \leq (\phi - \varepsilon)N \), since \( \tilde{f} \leq f \), item not reported
  - If \( f \geq \phi N \), then item is always reported because
    \[
    \tilde{f} \geq f - N\varepsilon \geq (\phi - \varepsilon)N
    \]
External-Memory Misra Gries

Structure

- A sequence of geometrically increasing Misra-Gries tables
- The smallest table is in memory and is of size $M$, the last table is of size $\lceil 1/\varepsilon \rceil$
- Total levels $= O(\log(1/\varepsilon M))$

Algorithm

- The top level receives its input from the stream
- Decrements from one level are inputs to the level below
- Decrements from the last level leave the structure
External-Memory Misra Gries

\[ Mr^L = \lceil 1/\epsilon \rceil \]
External-Memory Misra Gries

\[ M = \lceil \frac{1}{\epsilon} \rceil \]

\[ Mr^L = \lceil 1 / \epsilon \rceil \]
External-Memory Misra Gries

\[ Mr^L = \lceil 1/\varepsilon \rceil \]
External-Memory Misra Gries

\[ Mr^L = \left\lceil \frac{1}{\varepsilon} \right\rceil \]
External-Memory Misra Gries

\[ Mr^L = \lceil 1/\varepsilon \rceil \]

\[ Mr^2 \]

\[ Mr \]

\[ M \]

\[ \text{RAM} \]

\[ \text{Disk} \]
External-Memory Misra Gries

$M_r$  

$M_r^2$  

$M_r^L = \lceil 1/\epsilon \rceil$
External-Memory Misra Gries

Moving element counts from one level to another is a **flush**

Items flushed from the last level are deleted

\[ Mr^{L-1} = \lceil 1/\varepsilon \rceil \]
EM Misra Gries Analysis

**Theorem.** Amortized cost of insert in EM Misra Gries is $O\left( \frac{1}{B} \log \frac{1}{\epsilon M} \right)$

- A flush from level $i$ and inserting into level $i + 1$ costs $O\left( \frac{r^{i+1}M}{B} \right)$ I/Os
- Each such *flush* moves $r^i M$ down one level
- Amortized cost of a flush $= \frac{r^{i+1}M}{r^i M} \cdot \frac{1}{B} = O\left( \frac{r}{B} \right)$
- Each element can move down at most $\log 1/(\epsilon M)$ levels
External-Memory MG Algorithm: Takeaways

- **Supports fast inserts for small $\varepsilon$.**
  For the common case, when $B = \Omega(\log N)$, the cost of inserting into an external-memory MG algorithm even for small $\varepsilon$ is $<< 1$ I/O.

- **Does not support timely reporting.**
  Counts of items may be buried on lower levels on disk, that is, online event detection is no longer possible
Towards Online Event Detection

How do we get timely reporting?

- **OEDP.** We can pay for more I/Os to do queries
  - When to query? Querying on every insert is too expensive

- **OEDP with time stretch.** No explicit queries necessary if bounded delay is allowed
  - The algorithm can “organically” find the events
OEDP Algorithm

- Modify external-memory MG algorithm to support timely reporting
- When the in-memory count estimate of an item reaches the reporting threshold of RAM MG table \( (\phi - 1/M)N \), query all levels for rest of the counts
- If consolidated count reaches overall threshold \( (\phi - \epsilon)N \) then report
OEDP Algorithm

Suppose $N/M = 31$, $\phi N = 40$, $\epsilon N = 3$

Reporting threshold in RAM: $(\phi - 1/M)N = 9$

Report if total count reaches $\phi N - \epsilon N = 27$

$Mr^L = \lceil 1/\epsilon \rceil$
OEDP Algorithm Analysis

**Theorem.** Given a stream of size $N$, and $\phi > 1/M + \Omega(1/N)$, the amortized cost of solving OEDP is $O\left(\left(\frac{1}{B} + \frac{1}{(\phi - 1/M)N}\right) \log \frac{1}{eM}\right)$.
Theorem. Given a stream of size $N$, and $\phi > 1/M + \Omega(1/N)$, the amortized cost of solving OEDP is

$$O\left(\left(\frac{1}{B} + \frac{1}{(\phi - 1/M)N}\right) \log \frac{1}{\epsilon M}\right)$$

Fraction of elements that have count $\geq (\phi - 1/M)N$
Theorem. Given a stream of size $N$, and $\phi > 1/M + \Omega(1/N)$, the amortized cost of solving OEDP is 

$$O\left(\left(\frac{1}{B} + \frac{1}{(\phi - 1/M)N}\right) \log \frac{1}{\epsilon M}\right)$$

This can be very expensive if $\phi$ is close to $1/M$!
Bounded Reporting Delay

OEDP with Time Stretch

For a time-stretch of $1 + \alpha$, we must report an element a no later than time $t_1 + (1 + \alpha)F_t$

Key idea: the longer the flow time of a key, the more leeway we have in reporting it.
Time-Stretch Filter

- Cascade of geometrically increasing tables
- Total levels = $O(\log N/M)$

\[
\begin{align*}
\text{RAM} & \quad M \\
\text{EM} & \quad Mr \\
\text{EM} & \quad Mr^2 \\
\text{EM} & \quad Mr^L = N
\end{align*}
\]
Time-Stretch Filter

Divide each level into \((1 + 1/\alpha)\) equal-sized bins

\[ \frac{1}{\alpha} \text{ bins} \]

\[ M \]

RAM

EM

\[ Mr \]

\[ Mr^i \]

\[ Mr^{i+1} \]
Time-Stretch Filter

When a bin is full, items move to adjacent bin

1/\alpha \text{ bins}

RAM

EM

Mr

Mr^i

Mr^{i+1}
Time-Stretch Filter

When a bin is full, items move to adjacent bin

\[ \frac{1}{\alpha} \text{ bins} \]

\[ M \]

\[ M_r \]

\[ M_r^i \]

\[ M_r^{i+1} \]
Time-Stretch Filter

When a bin is full, items move to adjacent bin

\[ \frac{1}{\alpha} \text{ bins} \]

\[ M \]

\[ \text{RAM} \]

\[ EM \]

\[ M_r \]

\[ M_r^i \]

\[ M_r^{i+1} \]
Time-Stretch Filter

When a bin is full, items move to adjacent bin

$1/\alpha$ bins

$M$

RAM

EM

$Mr$

$Mr^i$

$Mr^{i+1}$
Time-Stretch Filter

Last bin flushed to first bin of next level

1/\alpha \text{ bins}

\begin{align*}
M_r^i & \quad M_r^{i+1} \\
\text{RAM} & \quad \text{EM}
\end{align*}
Time-Stretch Filter

While flushing consolidate all counts; report if hits threshold

Last bin **flushed** to first bin of next level

While flushing consolidate all counts; report if hits threshold
Time-Stretch Filter

\[ \frac{1}{\alpha} \text{ bins} \]

Last bin **flushed** to first bin of next level

\[ M \]

\[ Mr \]

\[ Mr^i \]

\[ Mr^{i+1} \]
Main idea: key is not put on a deeper level until it has "aged sufficiently"
Time-Stretch Filter Correctness

Let $i + 1$ be the lowest level a key is at when it hits the threshold count.

\[ \frac{1}{\alpha} \text{ bins} \]

\[ \frac{1}{\alpha} \text{ bins of size } \frac{\alpha}{\alpha + 1} \cdot r^i M \]
Let $i + 1$ be the lowest level a key is at when it hits the threshold count.

Must have waited $1/\alpha$ bins at each level up to $i$ since its first arrival, dominated by wait at $i$.
Let $i + 1$ be the lowest level a key is at when it hits the threshold count.

Must have waited $1/\alpha$ bins at each level up to $i$ since its first arrival, dominated by wait at $i$.

That is,

$$F_t \geq \frac{r^i M}{\alpha + 1}$$
Let $i + 1$ be the lowest level a key is at when it hits the threshold count.

Must have waited $\frac{1}{\alpha}$ bins at each level up to $i$ since its first arrival, dominated by wait at $i$.

That is,

$$F_t \geq \frac{r^i M}{\alpha + 1}$$

Level $i + 1$ will participate in a flush again in

$$\frac{\alpha r^i M}{\alpha + 1} \leq \alpha F_t$$

time steps—key will be reported.
Theorem. Given a stream of size $N$, the amortized cost of solving OEDP with a time stretch $1 + \alpha$ is $O\left(\left(\frac{1 + \alpha}{\alpha}\right)\frac{1}{B} \log \frac{N}{M}\right)$. 

Optimal insert cost for EM & write-optimized dictionaries
Theorem. Given a stream of size $N$, the amortized cost of solving OEDP with a time stretch $1 + \alpha$ is 

$$O\left(\left(\frac{1 + \alpha}{\alpha}\right) \frac{1}{B} \log \frac{N}{M}\right)$$

Factor lost because we only flush a fraction of each level;
Constant loss for constant $\alpha$

Almost-online reporting with no extra query cost!
Implementation/ Optimizations

- Counts stored succinctly using **counting quotient filters** (CQFs) [Pandey et al. 17]
- Deamortize by dividing filter at each level into multiple smaller filters called *cones*
Implementation/ Optimizations

- Multi-threaded implementation
- Each thread operates by first taking a lock at the cone and then performing the insert operation
Implementation/ Optimizations

- Multi-threaded implementation
- Each thread operates by first taking a lock at the cone and then performing the insert operation
- If there is contention, the thread then inserts the item in its local buffer and continues
Evaluations: Time Stretch

- Time-stretch filter gives improvements in timely reporting compared to out-of-the-box structures like cascade filters [Bender et al.12]
The EM MG filter without immediate reporting has the highest throughput, followed by other variants.
Evaluations: Scalability

- With higher # of threads and higher N/M, the counter-stretch filter beats throughput of regular MG algorithm
Evaluations: I/O Performance

- The total I/O measured using ‘iotop’ and calculated theoretically is similar for various versions of the OEDP data structures.
Conclusions & Future Directions

• Bridging the gap between streaming & external memory
• With modern SSDs and I/O techniques possible to match cache-latency-bound in-memory data structures in EM
• What other streaming problems can be solved exactly in EM at comparable speed?
• What is the write model for streaming in modern EM?
Tenure-Track Faculty Position in Computer Science

Williams College: Massachusetts: Computer Science

Location: Williamstown, MA  
Open Date: Aug 8, 2019

Description:
The Department of Computer Science at Williams College invites applications for a tenure-track position at the rank of assistant professor beginning fall 2020. In an exceptional case, a more advanced appointment may be considered. The position has a three-year initial term and is open to all areas of computer science. We are especially interested in candidates with strong backgrounds in Machine Learning, Artificial Intelligence, Natural Language Processing, or Computer Graphics, but applicants from all areas are encouraged to apply.

All applications received by December 1 will receive full consideration, and review of applications will continue until the position is filled.

Application Process:
This institution is using Interfolio’s Faculty Search to conduct this search. Applicants to this position receive a free Dossier account and can send all application materials, including confidential letters of recommendation, free of charge.

Apply Now
The Jobs I Didn’t See: My Misconceptions of the Academic Job Market
THANK YOU