Streaming Algorithms

Stony Brook University
CSE545, Fall 2016

Big Data Analytics -- The Class

We will learn:

- to analyze different types of data:
  - high dimensional
  - graphs
  - infinite/never-ending
  - labeled
- to use different models of computation:
  - MapReduce
  - streams and online algorithms
  - single machine in-memory
  - *Spark*

Big Data Analytics -- The Class

We will learn:

- to analyze different types of data:
  - high dimensional
  - graphs
  - **infinite/never-ending**
  - labeled
- to use different models of computation:
  - MapReduce
  - **streams and online algorithms**
  - single machine in-memory
  - *Spark*

Motivation

One often does not know when a set of data will end.

- Can not store
- Not practical to access repeatedly
- Rapidly arriving
- Does not make sense to ever “insert” into a database

Can not fit on disk but would like to generalize / summarize the data?

Motivation

One often does not know when a set of data will end.

- Can not store
- Not practical to access repeatedly
- Rapidly arriving
- Does not make sense to ever “insert” into a database

Can not fit on disk but would like to generalize / summarize the data?

Examples: Google search queries

Satellite imagery data

Text Messages, Status updates

Click Streams

Stream Queries

1. Standing Queries: Stored and permanently executing.

2. Ad-Hoc: One-time questions
   -- must store expected parts / summaries of streams

Stream Queries

1. Standing Queries: Stored and permanently executing.

2. Ad-Hoc: One-time questions
   -- must store expected parts / summaries of streams

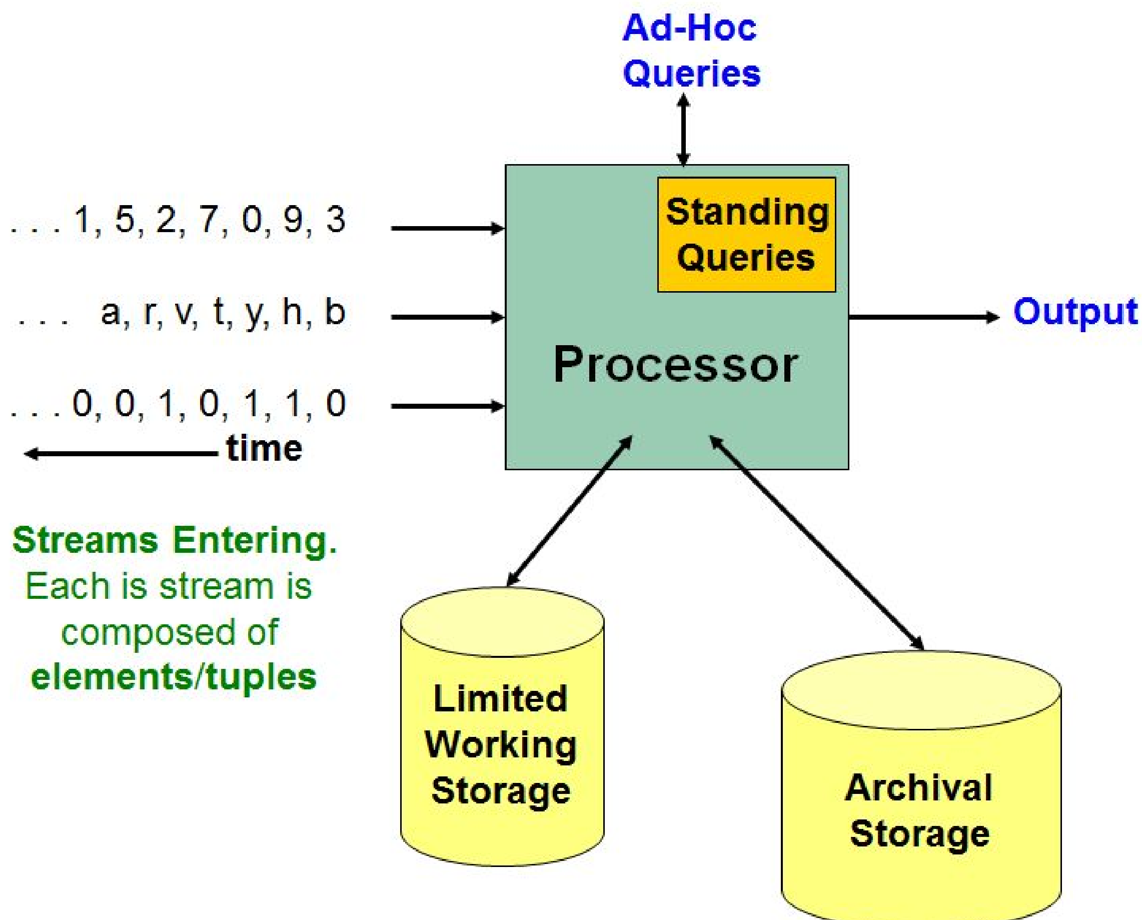
E.g. Each would handle the following differently:

What is the mean of values seen so far?

Streaming Algorithms

- Sampling
- Filtering Data
- Count Distinct Elements
- Counting Moments
- Incremental Processing*

General Stream Processing Model



Sampling and Filtering Data

Sampling: Create a random sample for statistical analysis.

- Basic version: generate random number; if < sample% keep
  - Problem: Tuples usually are not units-of-analysis for statistical analyses
- Assume provided some key as unit-of analysis to sample over
  - E.g. ip_address, user_id, document_id, ...etc….
- Want 1/20th of all “keys” (e.g. users)
  - Hash to 20 buckets; bucket 1 is “in”; others are “out”
  - Note: do not need to store anything (except hash functions); may be part of standing query

Sampling and Filtering Data

Filtering: Select elements with property x

Example: 40B email addresses to bypass spam filter

Sampling and Filtering Data

Filtering: Select elements with property x

Example: 40B email addresses to bypass spam filter

- The Bloom Filter
  - Given:
    - |S| keys to filter; will be mapped to |B| bits
    - hashes = $h_1, h_2, \ldots, h_k$ independent hash functions

Sampling and Filtering Data

Filtering: Select elements with property x

Example: 40B email addresses to bypass spam filter

- The Bloom Filter (approximates; allows FPs)
  - **Given**:
    - |S| keys to filter; will be mapped to |B| bits
    - hashes = $h_1, h_2, \ldots, h_k$ independent hash functions
  - **Algorithm**
    - Set all B to 0
    - For each i in hashes, for each s in S:

      Set B[h_i (s)] = 1

  … #usually embedded in other code
    - while key x arrives next in stream
      - if B[h_i (s)] == 1 for all i in hashes: do as if x is in S
      - else: do as if x not in S

Sampling and Filtering Data

Filtering: Select elements with property x

Example: 40B email addresses to bypass spam filter

- The Bloom Filter (approximates; allows FPs)
  - **Given:**
    - |S| keys to filter; will be mapped to |B| bits
    - hashes = $h_1, h_2, \ldots, h_k$ independent hash functions
  - **Algorithm**
    - Set all B to 0
    - For each i in hashes, for each s in S:

      Set B[h_i (s)] = 1

… #usually embedded in other code

- while key x arrives next in stream
  - if B[h_i (s)] == 1 all i in hashes: do as if x is in S
  - else: do as if x not in S

What is the probability of a false-positive?

Sampling and Filtering Data

Filtering: Select elements with property x

Example: 40B email addresses to bypass spam filter

- The Bloom Filter (approximates; allows FPs)
  - **Given**:
    - |S| keys to filter; will be mapped to |B| bits
    - hashes = $h_1, h_2, \ldots, h_k$ independent hash functions
  - **Algorithm**
    - Set all B to 0
    - For each i in hashes, for each s in S:

      Set B[h_i(s)] = 1

  … #usually embedded in other code
    - while key x arrives next in stream
      - if B[h_i(s)] == 1 all i in hashes: do as if x is in S
      - else: do as if x not in S

What is the probability of a false-positive?

What fraction of |B| are 1s?

Like throwing |S| * k darts at n targets.

1 dart: 1/n

D darts: $(1 - 1/n)^d$

$= e^{-d/n}$ faction are 1s

Sampling and Filtering Data

Filtering: Select elements with property x

Example: 40B email addresses to bypass spam filter

- The Bloom Filter (approximates; allows FPs)
  - **Given:**
    - |S| keys to filter; will be mapped to |B| bits
    - hashes = $h_{1,}$ h_{2}, …, h_k independent hash functions
  - **Algorithm**
    - Set all B to 0
    - For each i in hashes, for each s in S:

      Set B[h_i (s)] = 1

    … #usually embedded in other code
    - while key x arrives next in stream
      - if B[h_i (s)] == 1 all i in hashes: do as if x is in S
      - else: do as if x not in S

What is the probability of a false-positive?

What fraction of |B| are 1s?

Like throwing |S| * k darts at n targets.
1 dart: 1/n
D darts: $(1 - 1/n)^d$
= $e^{-d/n}$ faction are 1s

probability all k hashes being 1?

Sampling and Filtering Data

Filtering: Select elements with property x

Example: 40B email addresses to bypass spam filter

- The Bloom Filter (approximates; allows FPs)
  - **Given**:
    - |S| keys to filter; will be mapped to |B| bits
    - hashes = $h_1, h_2, \ldots, h_k$ independent hash functions
  - **Algorithm**
    - Set all B to 0
    - For each i in hashes, for each s in S:

      Set B[h_i (s)] = 1

… #usually embedded in other code

- while key x arrives next in stream
  - if B[h_i (s)] == 1 all i in hashes: do as if x is in S
  - else: do as if x not in S

What is the probability of a false-positive?

What fraction of |B| are 1s?

Like throwing |S| * k darts at n targets.
1 dart: 1/n
D darts: $(1 - 1/n)^d$
= $e^{-d/n}$ faction are 1s

probability all k hashes being 1?
$(e^{-(|S|*k)/n})^k$

Note: Can expand S as stream continues
(e.g. adding verified email addresses)

Counting Moments

Moments:

- Suppose m_i is the count of distinct element i in the data
- The kth moment of the stream is $\sum_{i \in \mathrm{Set}} m_i^k$

Examples

- 0th moment: count of distinct elements
- 1st moment: length of stream
- 2nd moment: sum of squares (measures *uneveness* related to variance)

Counting Moments

0th moment
One Solution: Just keep a set (hashmap, dictionary, heap)

Problem: Can’t maintain that many in memory; disk storage is too slow

- **0th moment: count of distinct elements**
- 1st moment: length of stream
- 2nd moment: sum of squares (measures *uneveness* related to variance)

Counting Moments

0th moment

Streaming Solution: Flajolet-Martin Algorithm

Pick a hash, *h,* to map each of n elements to $\log_2 n$ bits

R = 0 #potential max number of zeros at tail

for each stream element, e:

r(e) = num of trailing 0s from *h*(e)

R = r(e) if r(e) > R

estimated_distinct_elements = 2^R

- **0th moment: count of distinct elements**
- 1st moment: length of stream
- 2nd moment: sum of squares (measures *uneveness* related to variance)

Counting Moments

0th moment

Streaming Solution: Flajolet-Martin Algorithm

Pick a hash, *h,* to map each of n elements to $\log_2 n$ bits
R = 0 #potential max number of zeros at tail
for each stream element, e:
    r(e) = num of trailing 0s from *h*(e)
    R = r(e) if r(e) > R
estimated_distinct_elements = 2^R

Problem:
Unstable in practice.

- **0th moment: count of distinct elements**
- 1st moment: length of stream
- 2nd moment: sum of squares (measures *uneveness* related to variance)

Counting Moments

0th moment

Streaming Solution: Flajolet-Martin Algorithm

Pick a hash, *h,* to map each of n elements to $\log_2 n$ bits

R = 0 #potential max number of zeros at tail

for each stream element, e:

r(e) = num of trailing 0s from *h*(e)

R = r(e) if r(e) > R

estimated_distinct_elements = 2^R

Problem:
Unstable in practice.

Solution:

1. Partition into groups
2. Take mean in group
3. Take median of means

- **0th moment: count of distinct elements**
- 1st moment: length of stream
- 2nd moment: sum of squares (measures *uneveness* related to variance)

Counting Moments

1st moment
Streaming Solution: Simply keep a counter

- 0th moment: count of distinct elements
- **1st moment: length of stream**
- 2nd moment: sum of squares (measures *uneveness* related to variance)

Counting Moments

2nd moment
Streaming Solution: Alon-Matias-Szegedy Algorithm

(Exercise; see in MMDS)

- 0th moment: count of distinct elements
- 1st moment: length of stream
- **2nd moment: sum of squares (measures *uneveness* related to variance)**