**COMPSCI 590S Systems for Data Science** **Fall 2016**

# Lecture 14: Bloom Filter, Distributed Shared Memory and Spark

*Lecturer: Emery Berger* *Scribe(s): Namrita Pandita, Mark Saad*

## 14.1 Bloom Filter [Used in BigTable]

What is Bloom Filter?

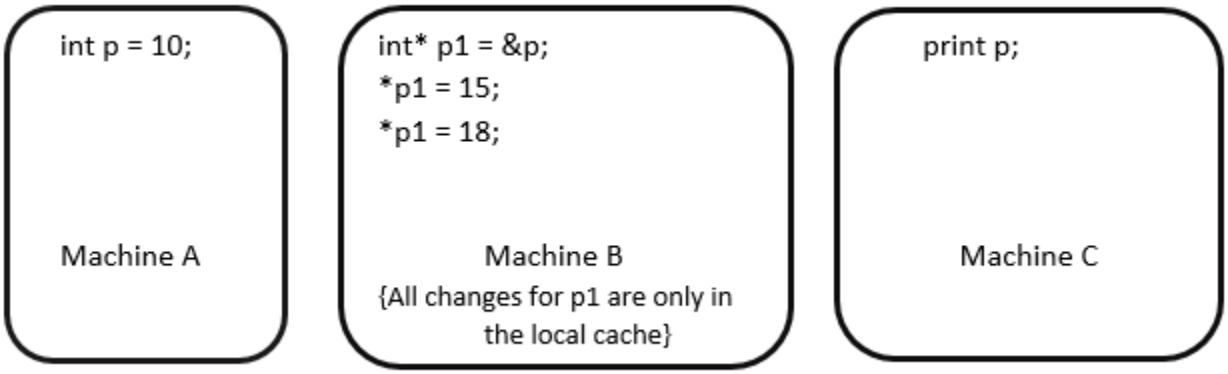
- tries to find if an element x is in set S
- has one way error meaning:
  - If it returns false then x definitely does not belong to S
  - If it returns true then we dont know! X may or may not belong to S

Why is it needed?

- And most cases are expected to false and returning false is really fast and reliable.
- Compact representation
  - For a set from 0 to 15
    - A set of 16 bits: 0 represents that the item is not in the set, 1 is in the set. This if you know the cardinality and its compact
  - Else, like if youre doing all the names in the US
    - Use hash map, or use bloom set for all Americans (first + last) name. basically a set of zeros and you OR in every new name.
    - examples:
      - check if Mickey is in the set -¿ hash(Mickey) = 11 or 1011 in binary
      - if current set is S = 0000000110 it returns false[ because all bits set in hash of Mickey are not set in S], therefore Mickey is not in the set
      - check if Molly is in the set -¿ hash(Molly) = 2 or 10 in binary
      - returns true, therefore Molly may or may not exist in the set
    - the hash takes everybody in the set and distributes them, ideally randomly
    - it is usually a good idea to have a big enough bloom set so that you have less collisions

## 14.2 Distributed Shared Memory

The idea is to partition memory [RAM] into machines in the cluster. Partitioning may be done equally or using hashing. DSM is very fine grained [has direct access to memory].




Issues:

- Cache consistency (who sees what when)
  - Professor Emery thinks this is never gonna work: you have memory, its close, its fast, the end
- Race Conditions
- Example:
  - Cache Inconsistency: Machine B thinks p = 18, but Machine C thinks it is still 10, value in machine A is stale
  - Race Conditions: if Reads and writes to the same memory location occur simultaneously, the results will never be deterministic.

## 14.3 Spark

Spark is an API on top of RDDs:

- coarse-grained: less message traffic, it means that it can work on several or all objects at the same time, instead of looping over objects to set a value. You call a method like setAll and it does it for you in parallel, the same for a map function. Spark has datasets and coarse-grained operators that applies to everything or several items at a time.
- Deterministic: you dont have to worry about locks. [No Race Conditions]
- Immutable: state cant be changed; the item is not being modified in place. it also means that caching is fine and its better for parallelism [No Cache Inconsistency]

Lineage: low disk bandwidth, immutable operations, can be replayed easily. Ensures Fault tolerance.

RDDs are read-only in the sense that its write-once actions or transformations that operate on RDDs and create new RDDs

Persist is a misused term, in Sparks world it means save to memory (RAM) instead of the real meaning of persist which means write to disk

If you dont call persist, RDDs will be garbage collected

Spilling: RDDs can be written on disk (HDFS) if theres no available memory.

Serialization translates an object to a JSON-like format to transfer it to other machines, in order to avoid collisions. Spark is de-serialized. It is faster but not so memory efficient. Serialized version is more compact and highly optimized but has slow iteration. Serialization is needed if Java objects need to be moved from one machine to another.

Narrow vs wide

- Narrow is an operation that just goes from one to one on the same machine like a map operation without a shuffle
- Wide is an operation like join where it has implications for lineage. Has a shuffle involved [one-to-many or many-to-many]

Sparks Design:

- Batch Analysis: focused on iteration and is coarse grained
- Is not interactive and write intensive transactional
- Spark can be stated as Scala implementation of DryadLINQ along with RDDs.