Towards an Understanding of Scalable Query and Data Analysis for Social Media Data using High-Level Dataflow Systems

Stephen Wu Judy Qiu
Indiana University

Acknowledgements

Stephen Wu Xiaoming Gao Bingjing Zhang

Prof. Filippo Menczer & CNETS
Complex Networks and Systems

SALSA HPC Group
School of Informatics and Computing
Indiana University

Microsoft Google NSF NATIONAL INSTITUTES OF HEALTH intel

Outline

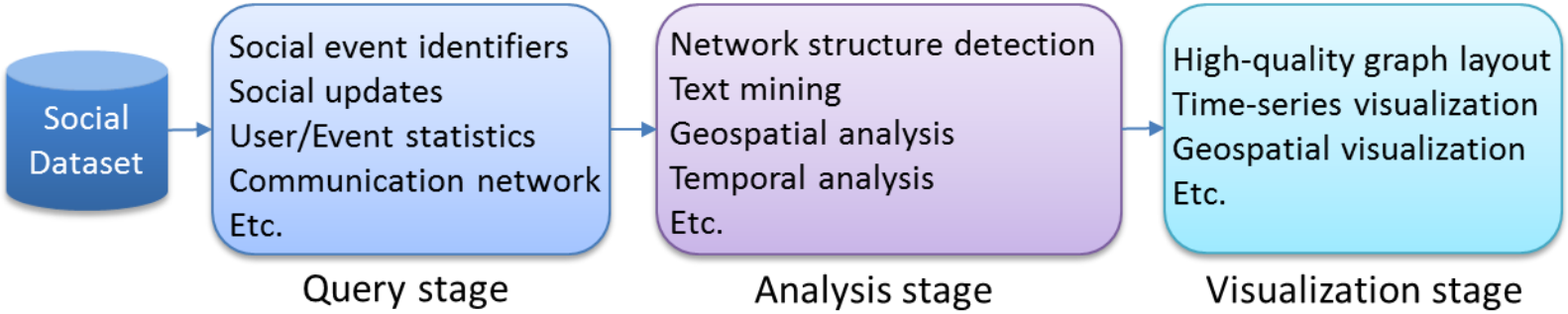
- Social media data and emerging characteristics
- Query using Pig and Hive with IndexedHBase
- Integration of query and analysis
- Summary and future work

Social media data - Emerging Characteristics

Velocity	Streaming data becoming more and more important	10s to 100s of millions of streaming social updates per day
Volume	Analyses focusing on “interesting” data subsets	Data subsets about social events/activities
Variety	Sensor data streams, stock price streams, etc. Gene sequence analysis, news data analysis, etc.	Social media data (e.g. Twitter streaming API)

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Social Media Data Analysis Process

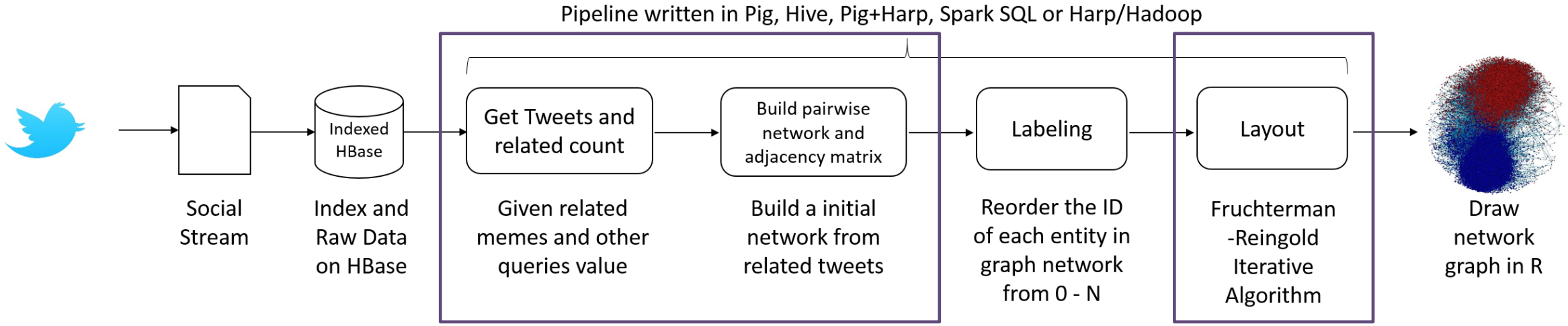


- Multiple stages of computations with different algorithms and runtimes
- Need manual/extra programming work, e.g. data formatting and ordering, for data between stages
- Social data size is large (from terabytes level to petabytes level)
  - E.g. Twitter collects more than 500 millions tweets per day and 200 billion tweets per year in 2013

*Borthakur, Dhruba, et al. "Apache Hadoop goes realtime at Facebook."*Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 2011.
*http://www.internetlivestats.com/twitter-statistics/

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

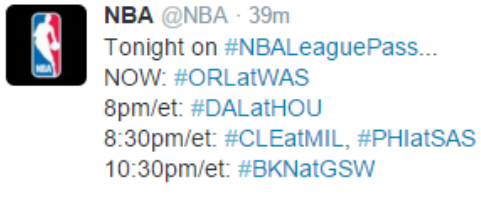
Truthy's Use Case: Visualizing Political Polarization



- Data collected from [Public Twitter Streaming API](#).
- Raw JSON Tweet Data and inverted indices are stored in HBase
  - Current data size as of Aug 2015 is approximately 162TB
- IndexedHBase API has provided load, index and query functions
- Initial queries are all written in IndexedHBase and serve as truthy-cmd.sh
  - Written in Java MapReduce, lack of high-level abstraction and flexibility to data scientists.

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

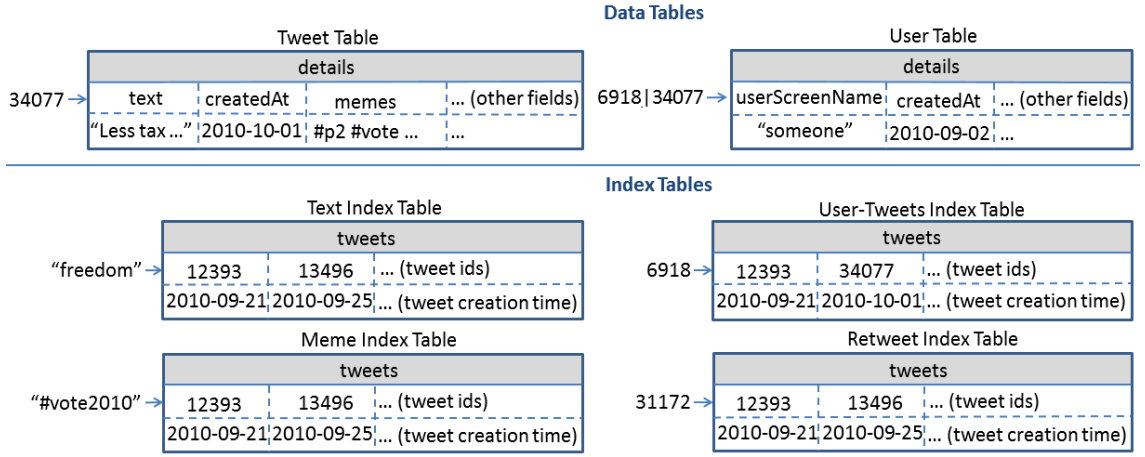
IndexedHBase (Data)



Tweet views on Twitter.com


```
{
    "text":"RT @sengineland: My Single Best... ",
    "created_at":"Fri Apr 15 23:37:26 +0000 2011",
    "retweet_count":0,
    "id_str":"59037647649259521",
    "entities":{
        "user_mentions":[{
                "screen_name":"sengineland",
                "id_str":"1059801",
                "name":"Search Engine Land",
            }],
        "hashtags":[],
        "urls":[{
                "url":"http:\/\/selnd.com\/e2QPS1",
                "expanded_url":null
            }]
    },
    "user":{
        "created_at":"Sat Jan 22 18:39:46 +0000 2011",
        "friends_count":63,
        "id_str":"241622902",
        ...
    },
    "retweeted_status":{
        "text":"My Single Best... ",
        "created_at":"Fri Apr 15 21:40:10 +0000 2011",
        "id_str":"59008136320786432",
                    ...
    },
    ...
}
```


Tweet published by Tweet Stream API



Data stored in HBase

- Convert tweets from JSON to Tables and store in HBase
- Fast and customized inverted index to search related tweet ID

** Images from IndexedHBase project website: http://salsaproj.indiana.edu/IndexedHBase/truthy.html*

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Definition of Ad hoc Query

- *"A query is a question you ask a database in the form of a defined format of statement (SQL or any other database syntax)"*
- *"Ad hoc means for this or for the purpose comes from Latin"*
- *"So, in database search, an ad hoc query is created to obtain information as the need arises. Contrast with a query that is predefined and routinely processed."*
- Our use cases of searching Twitter's tweets and indices stored in HBase are dynamically called by the users with specified interests of timeframes and user-defined fields (hashtags, keywords, username, word phrases, etc.)

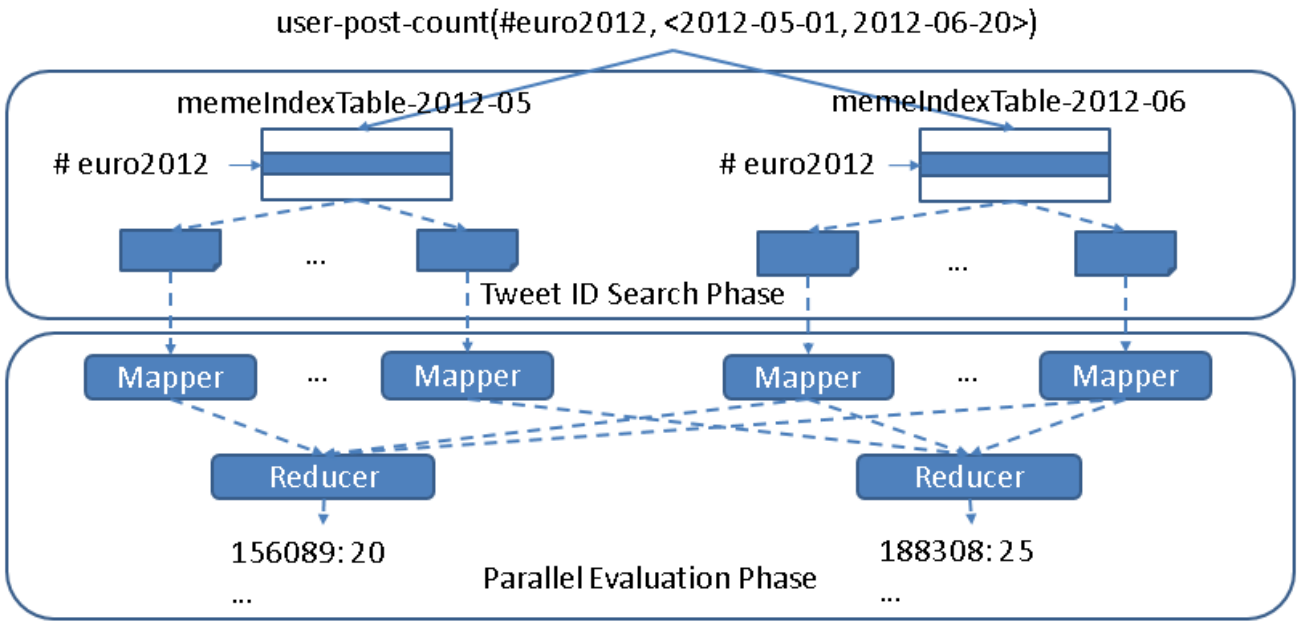
*http://www.answers.com/Q/What_is_ad_hoc_query
*https://en.wikipedia.org/wiki/Ad_hoc
*http://www.pcmag.com/encyclopedia/term/37486/ad-hoc-query

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

IndexedHBase (Queries)

Get tweet IDs by index

User Defined Functions
(Hadoop Mapper / UDF)



- Two separate parallel searches
  - Search related Tweet IDs by using a standalone multi-thread program
  - Use related Tweet IDs as input, submit and perform MapReduce job with UDF.
- Pig and Hive implementations follow this logic

** Images from IndexedHBase project website: http://salsaproj.indiana.edu/IndexedHBase/truthy.html*

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Comparison with Related Work

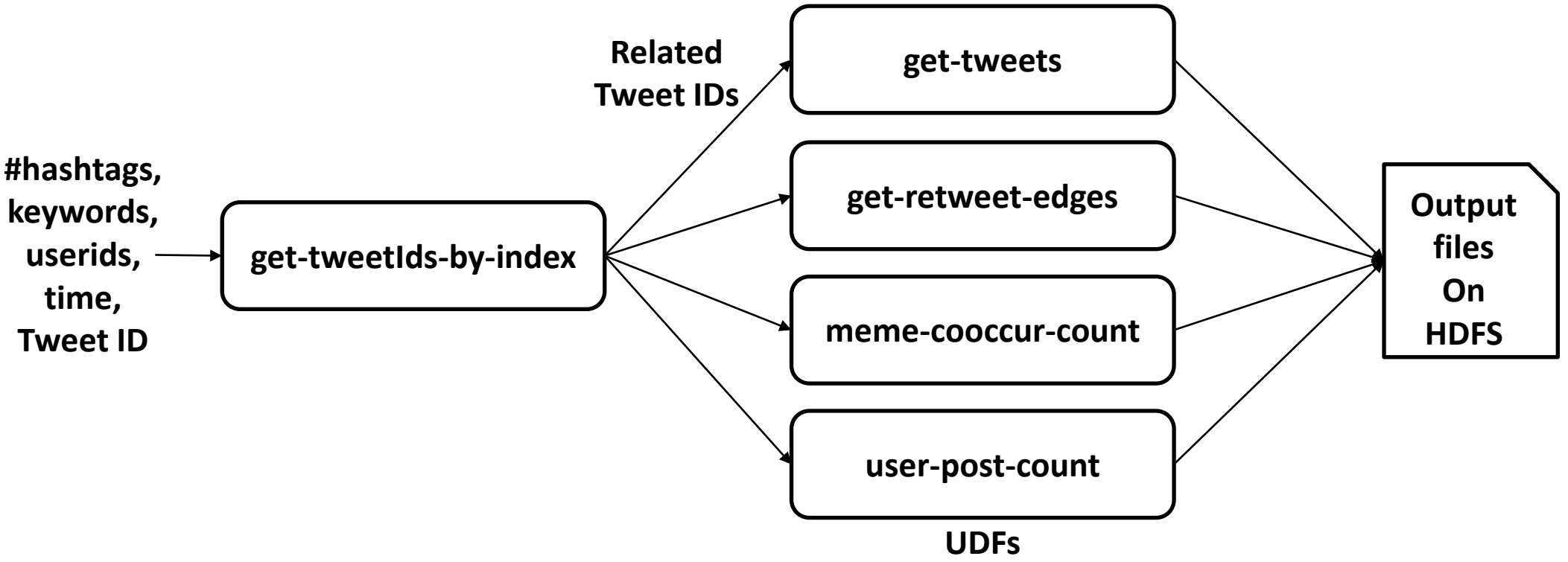
- Temporal-text queries, longitudinal analytics on web archives, etc.
- Online text indexing and incremental index maintenance
- Hadoop++, HAIL, Eagle-Eyed Elephant

Xiaoming Gao, Vaibhav Nachankar, Judy Qiu. Experimenting Lucene Index on HBase in an HPC Environment. Proc. 1st workshop on High-Performance Computing meets Databases (HPCDB 2011) at Supercomputing 2011.
Xiaoming Gao, Evan Roth, Karissa McKelvey, Clayton Davis, Andrew Younge, Emilio Ferrara, Filippo Menczer, and Judy Qiu. Supporting a Social Media Observatory with Customizable Index Structures - Architecture and Performance. Book chapter in Cloud Computing for Data Intensive Applications, Springer Publisher, 2015.

Outline

- Social media data and emerging characteristics
- Query using Pig and Hive with IndexedHBase
- Integration of query and analysis
- Summary and future work

Pig and Hive implementations



- Two separate Hadoop jobs
- UDFs use the tweet IDs as input, perform selected data transformation and yield different output to HDFS

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Supported Queries in Different Platforms

	Query	Queried key	IndexedHBase	Pig with IndexedHBaseStorage	Hive Default	Hive with IndexedHBaseStorage
1	get-tweets-with-meme	#ff	Y	Y	Y	Y
2	get-tweets-with-text	nba	Y	Y	Y	Y
3	get-tweets-with-userid	8401422	Y	Y	Y	Y
4	get-retweets	36128589241909248	Y	Y	Y	Y
5	get-tweets-with-time	2015-07-01, 2015-07-31	Y	Y	N	Y
6	(meme)-timestamp-count	#ff	Y	Y	N	Y
7	text-timestamp-count	nba	Y	Y	N	Y
8	userid-timestamp-count	8401422	Y	Y	N	Y
9	meme-post-count	#ff, #nba	Y	Y	Y	Y
10	text-post-count	nba	Y	Y	Y	Y
11	userid-post-count	2147485937 (BIGINT), 8401422 (INT)	Y	Y	Y	Y
12	user-post-count	#ff	Y	Y	Y	Y
13	user-post-count-by-text	nba	Y	Y	Y	Y
14	meme-cooccur-count	#ff	Y	Y	Y	Y
15	get-retweet-edges	#ff	Y	Y	Y	Y
16	get-mention-edges	#ff	Y	Y	Y	Y
17	get-tweets-with-phrase	yolo swag	Y	Y	Y	Y

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

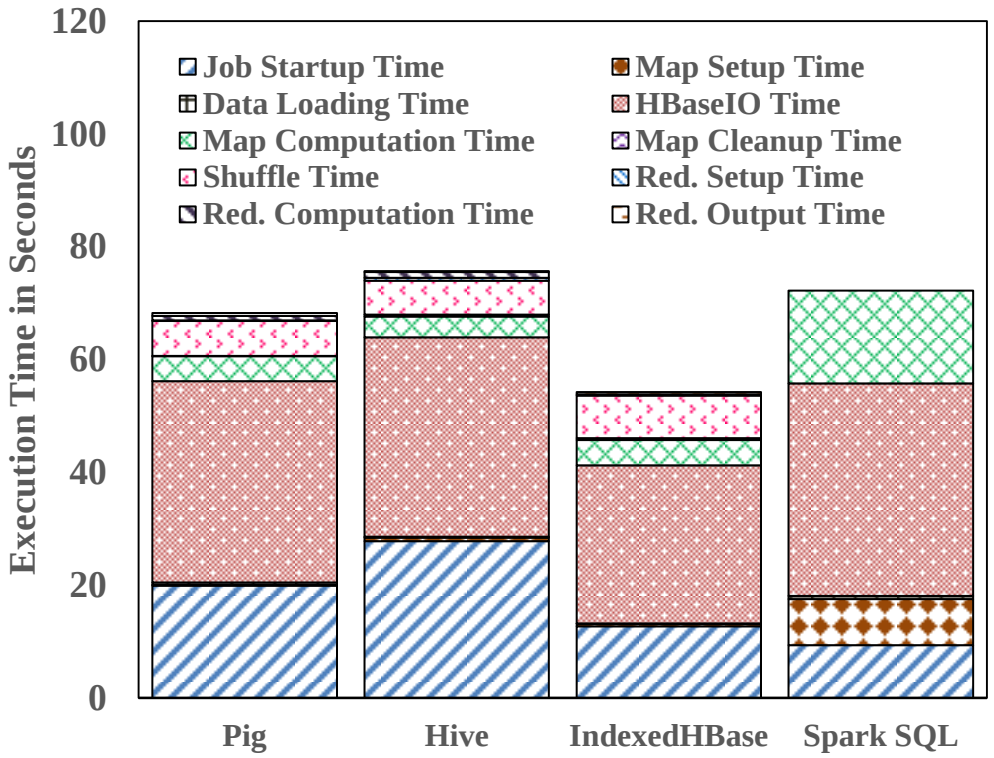
Use of General Operators in High-level Syntax

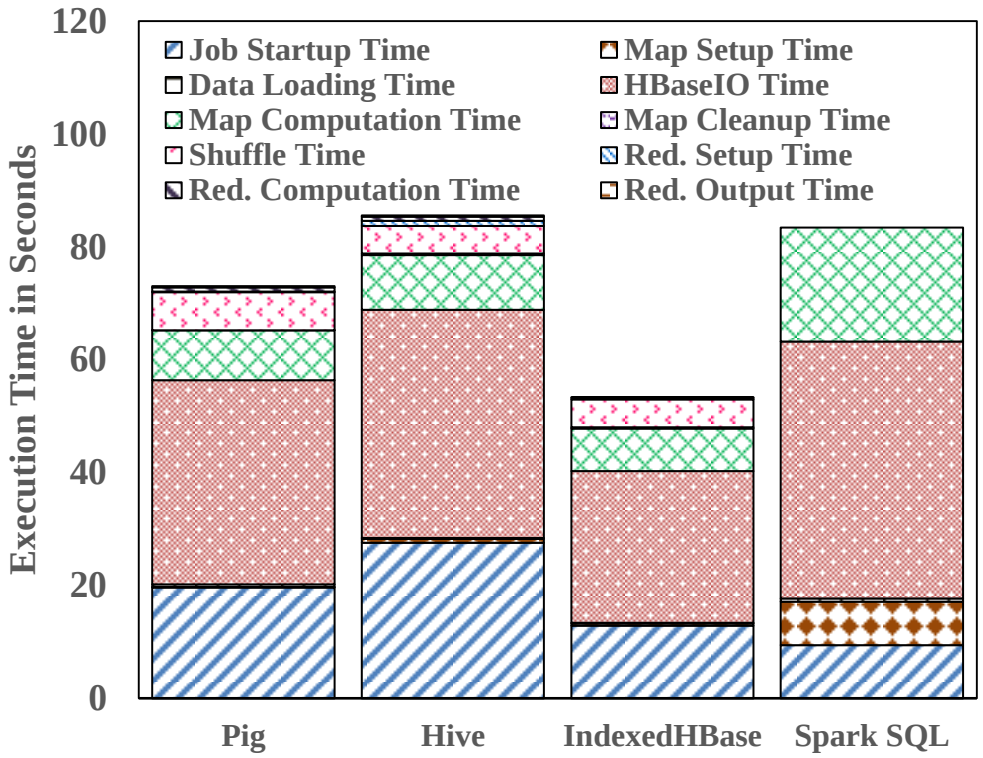
- Filter / Where
  - For queries such as #hashtags and keywords
- GROUP BY
  - Required by COUNT operator
- COUNT
  - Occurrence / Frequency
- CONCATENATE
  - Retweet/Mention EDGE, e.g. <userid> <retweet user id>
- NOT NULL

Pig and Hive Challenges

- Default Storage Handlers are limited
  - Only one queried key a time for random access
  - Filter (Where) operator with two keys in Hive is extremely slow as it scans the whole table
  - Lack of timestamp support in Hive
- We write an IndexedHBaseStorage API used for Pig and Hive
  - Support multiple queried keys
  - Support cross-month search to different table in single function call
  - Embedded table name in the output of “get-tweetIds-by-index”
    - E.g. tweetTable-2015-07 <tweet id>
- Searching in Truthy data has the following characteristics
  - Network I/O’s for reading a record from Tables in HBase is very fast
    - Obtain 328 records/second
  - Execution time of general ad-hoc queries is dominated by the size of related tweet IDs

Get-retweets-edges & Meme-co-occur-count



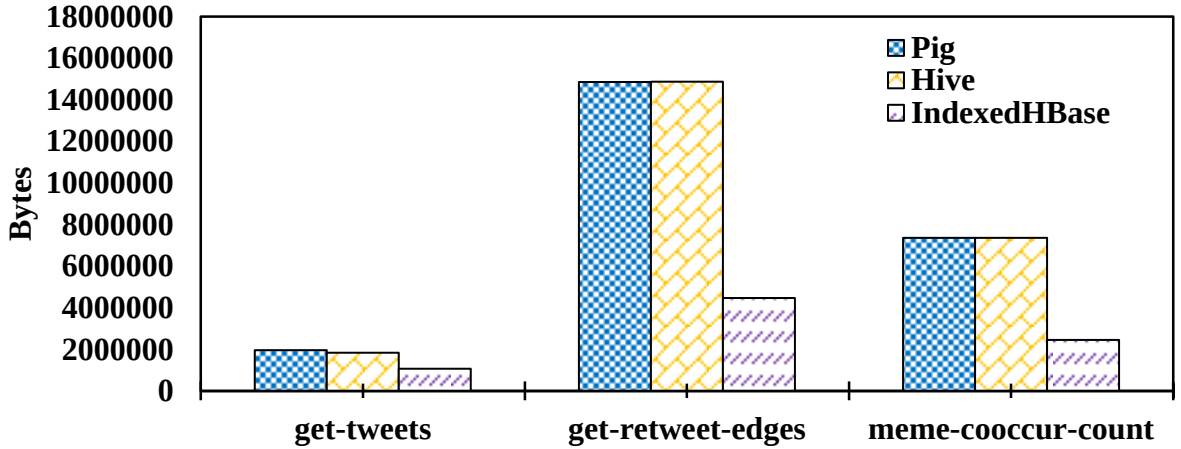


- Spark SQL handles computation and data aggregation with same set of containers
  - UDF transformation time (from DataFrameRDD to JavaRDD) and cross-worker data aggregation/communication time, along with the output to HDFS time

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Local Write & Intermediate Record Size

- Local write behaviors also prove IndexedHBase has lower I/O with the optimized combine at the end of the map cycle.



Query	Pig	Hive	IndexedHBase
get-tweets	587858	587858	587858
get-retweet-edges	179486	179463	167740
meme-cooccur-count	90216	90125	63524

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Execution Flow

- Truthy ad-hoc queries are very simple and straightforward; they do not have complicated execution logic.
  - Pig and Hive MapReduce job execution flow are similar

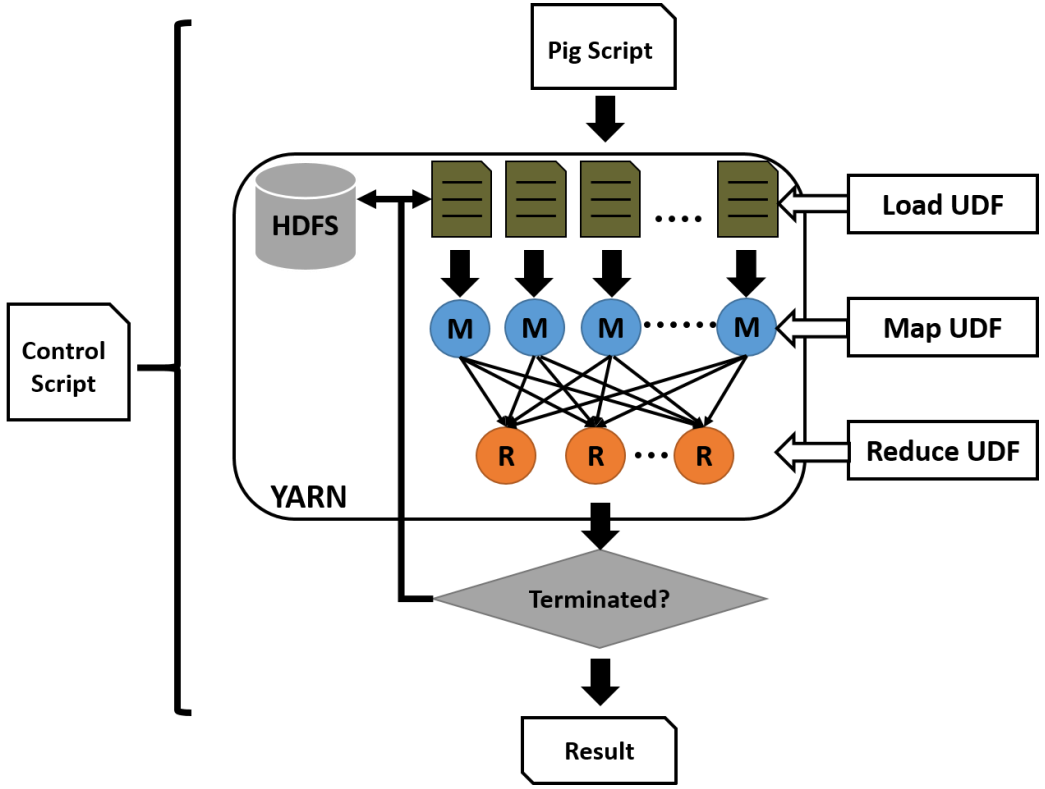
- Unless we work on the end-to-end solution with advanced post-query analysis, we can't make improvements on the query optimization.
  - Cost-based optimizations are mainly for select-project-join queries that read a lot from Tables, especially for filter operators
  - Use Apache Tez with session and reuse the containers for similar data

Outline

- Social media data and emerging characteristics
- Query using Pig and Hive with IndexedHBase
- Integration of query and analysis
- Summary and future work

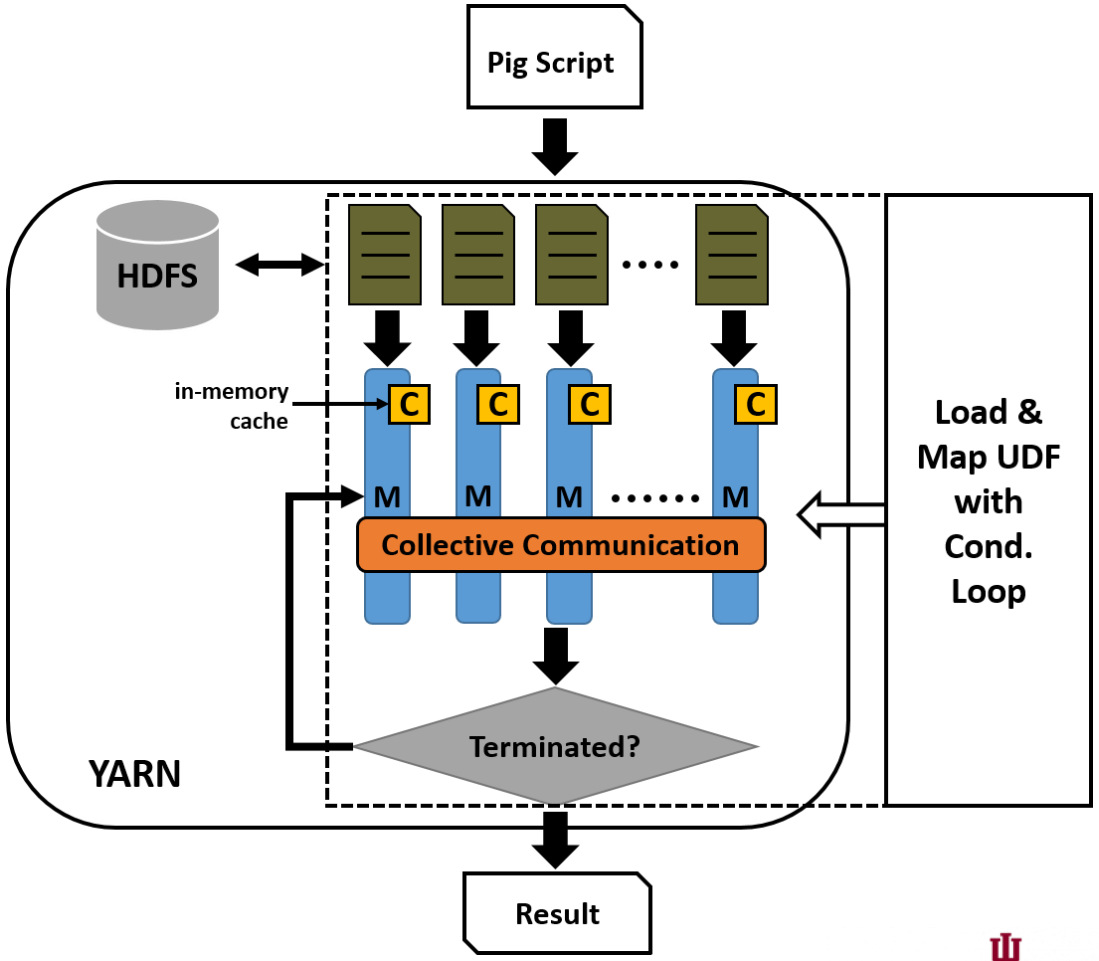
Pig and Iterative Applications

- Need a wrapper program to support conditional loop
- Intermediate results of iterations are mapped from disk to next iteration
  - Disk cache and Disk I/O are substantial
- Hadoop jobs restart overhead
- No in-memory caching mechanism
- Data partitions are based on Pig Input Format
- Tuple-based data transformation/computation is slow



Solution: Pig+Harp

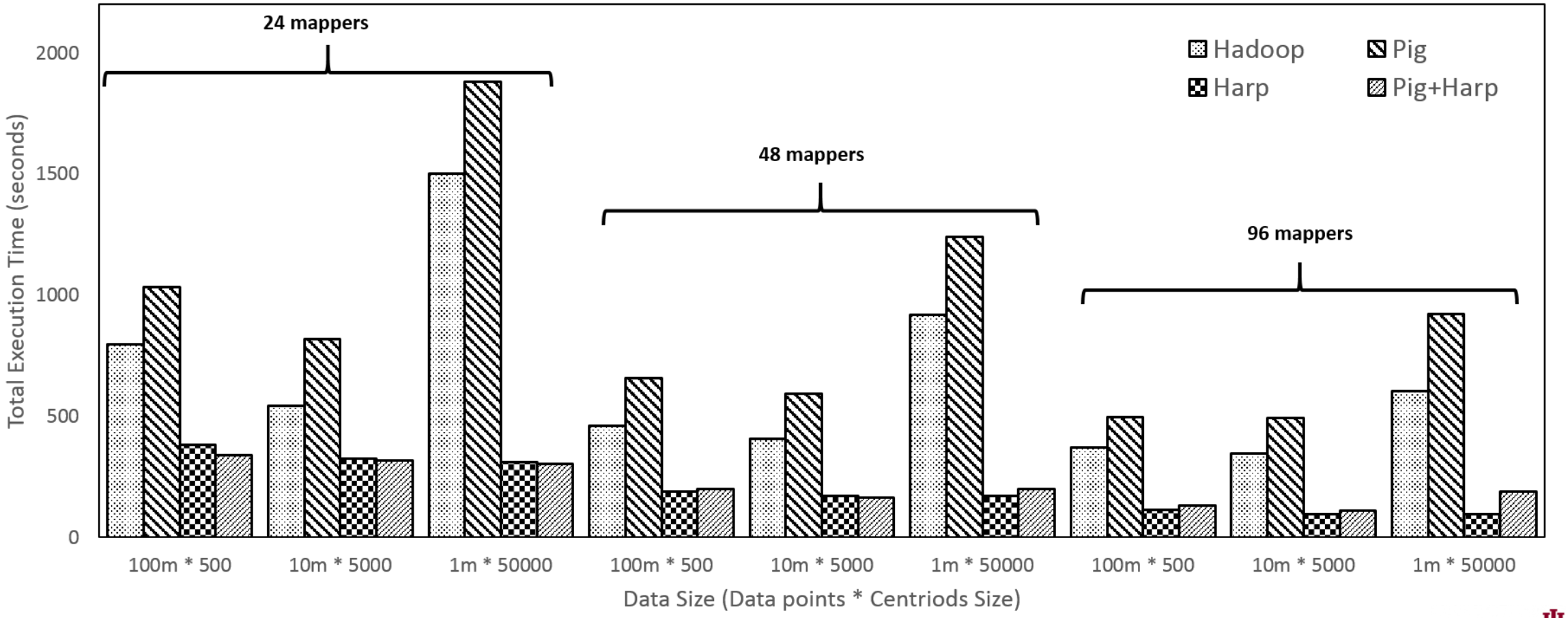
- Replace default mapper interface with Harp's MapCollective long-running mapper
- Read once, Compute many
- In-memory objects caching in LOAD & MAP stages' UDF
- Shuffle data by calling Harp's collective communication API
- UDF controls loop termination
- No-hassle plugins
  - Same as general Pig if collective communication is not written in UDF



** Tak Lon Wu, Abhilash Koppula, Judy Qiu Integrating Pig with Harp to Support Iterative Applications with Fast Cache and Customized Communication 5th International Workshop on Data Intensive Computing in the Clouds (DataCloud) 2014*

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

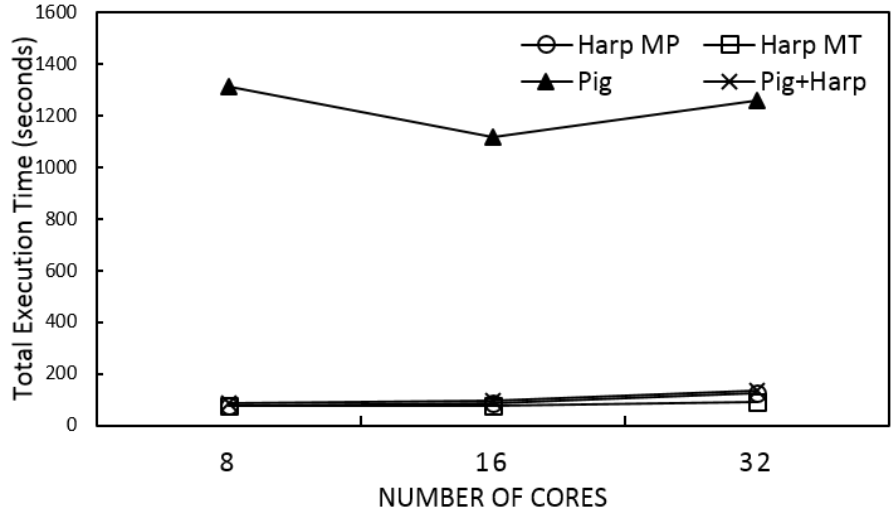
K-means Performance

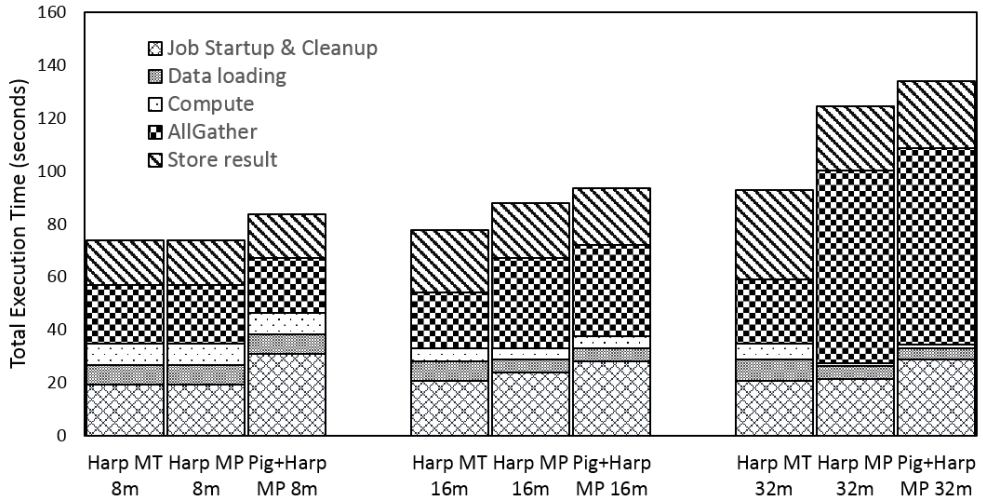


SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

PageRank

- Pig+Harp is 5 times faster than native Pig
  - Tuple-based computation
  - Data type conversion time between bags and fields
- Harp's multi-thread shows the advantage in AllGather communication for larger partitions
  - 2 layer synchronization
  - In-node sync and cross-node sync





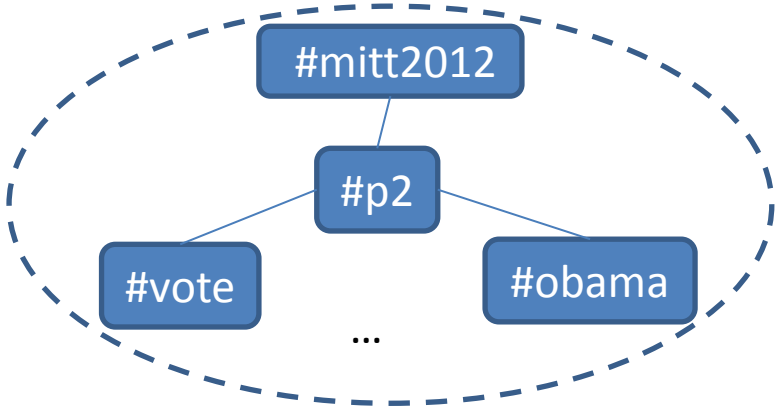
SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Analysis algorithms for composing workflows

Algorithm	Key feature	Time complexity
Related hashtag mining	Mostly relies on index; only accesses a small portion of original data.	O(H*M + N).
Meme daily frequency generation	Totally based on parallel scan of customized index.	O(N).
Domain name entropy computation	Totally based on parallel scan of customized index.	O(N).
Graph layout	First parallel implementation on iterative MapReduce; near-linear scalability.	$O(I*N^2)$.

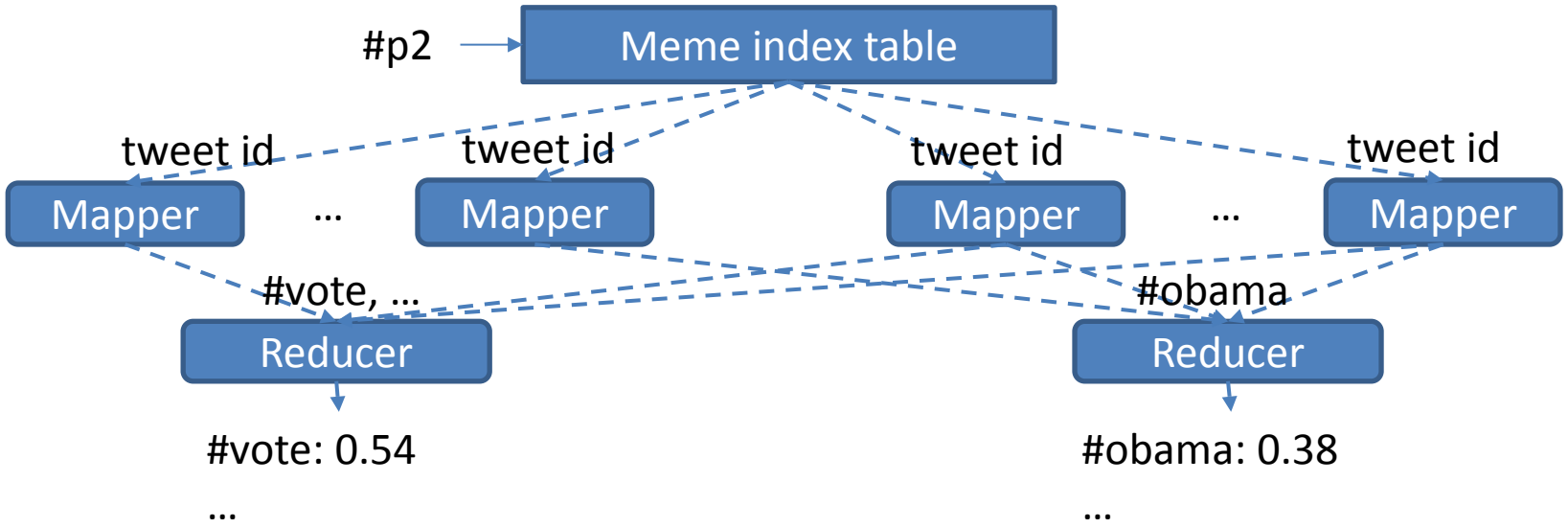
SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Related Hashtag mining



2012 presidential election

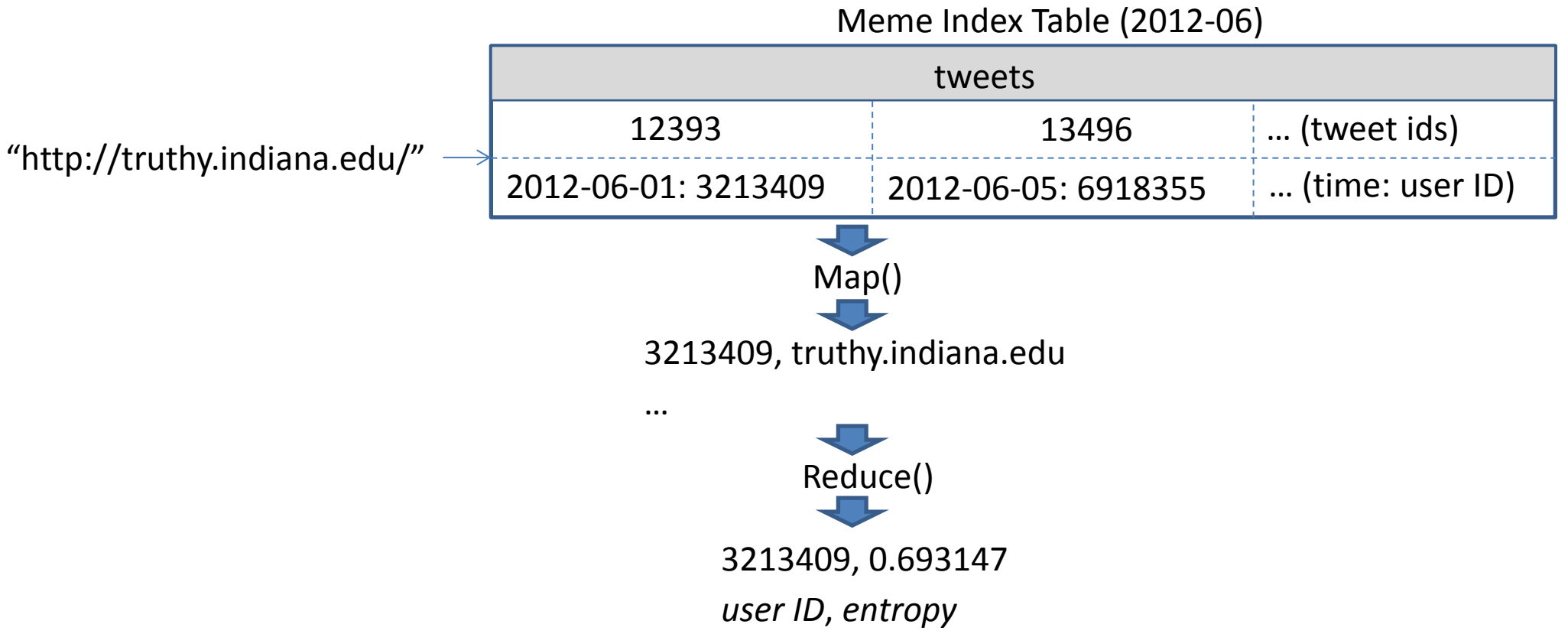
- Jaccard coefficient: $\sigma(S,T) = \frac{|S \cap T|}{|S \cup T|}$
- S: set of tweets containing seed hashtag *s*
- T: set of tweets containing target hashtag *t*
- σ > threshold means *t* is related to *s*



SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Domain name entropy computation

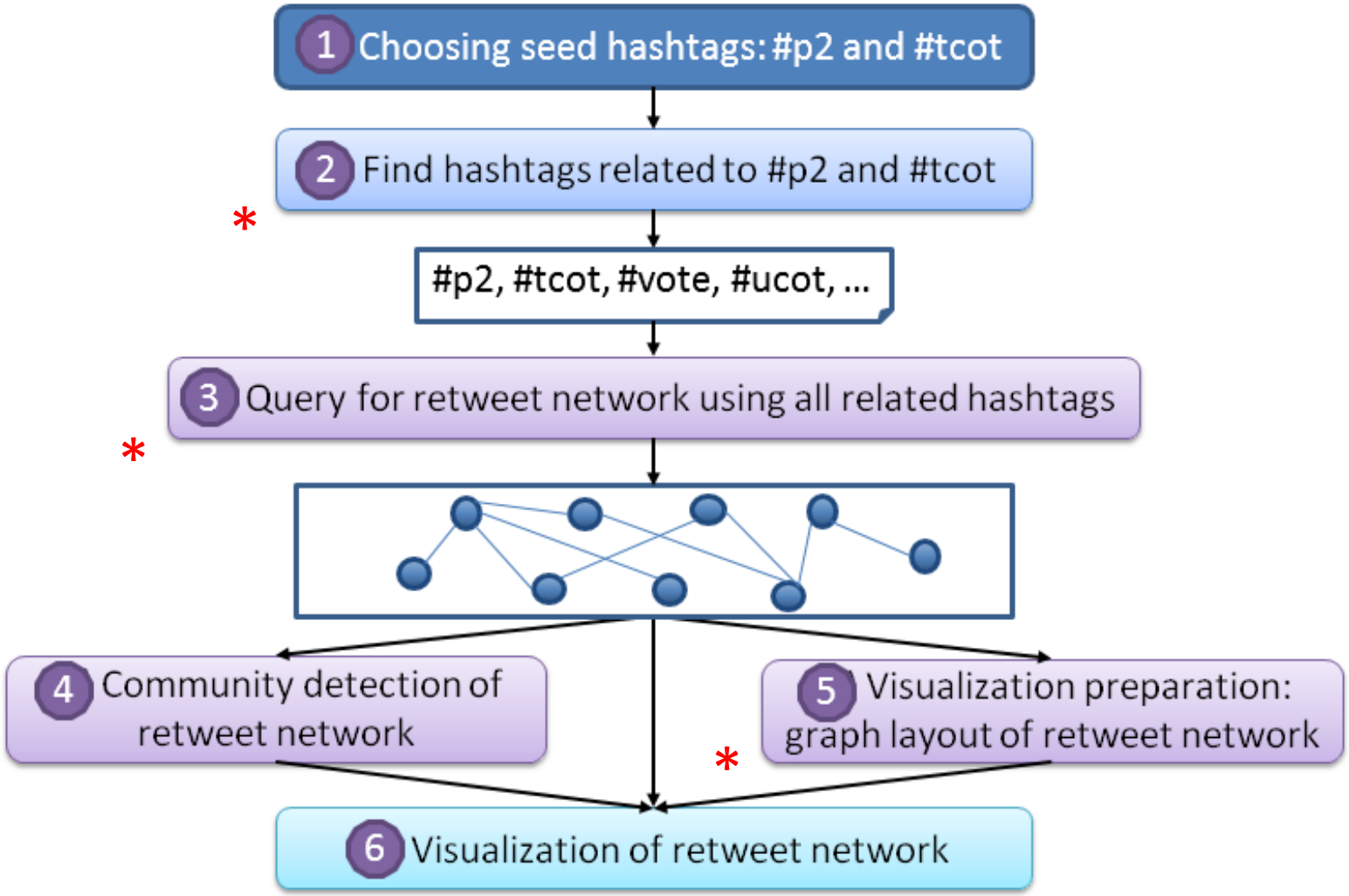
- For each user, find the domain names posted during certain time
- Compute entropy based on the domain name distribution



Force-directed graph layout algorithm

- Iterative MapReduce implementation of Fruchterman-Reingold
  - force-directed graph layout algorithm, complexity $O(I * N^2)$
  - Twister-Ivy (now Harp)
  - parallel force computation within iteration
  - chain model broadcast across iteration

Composition of workflows

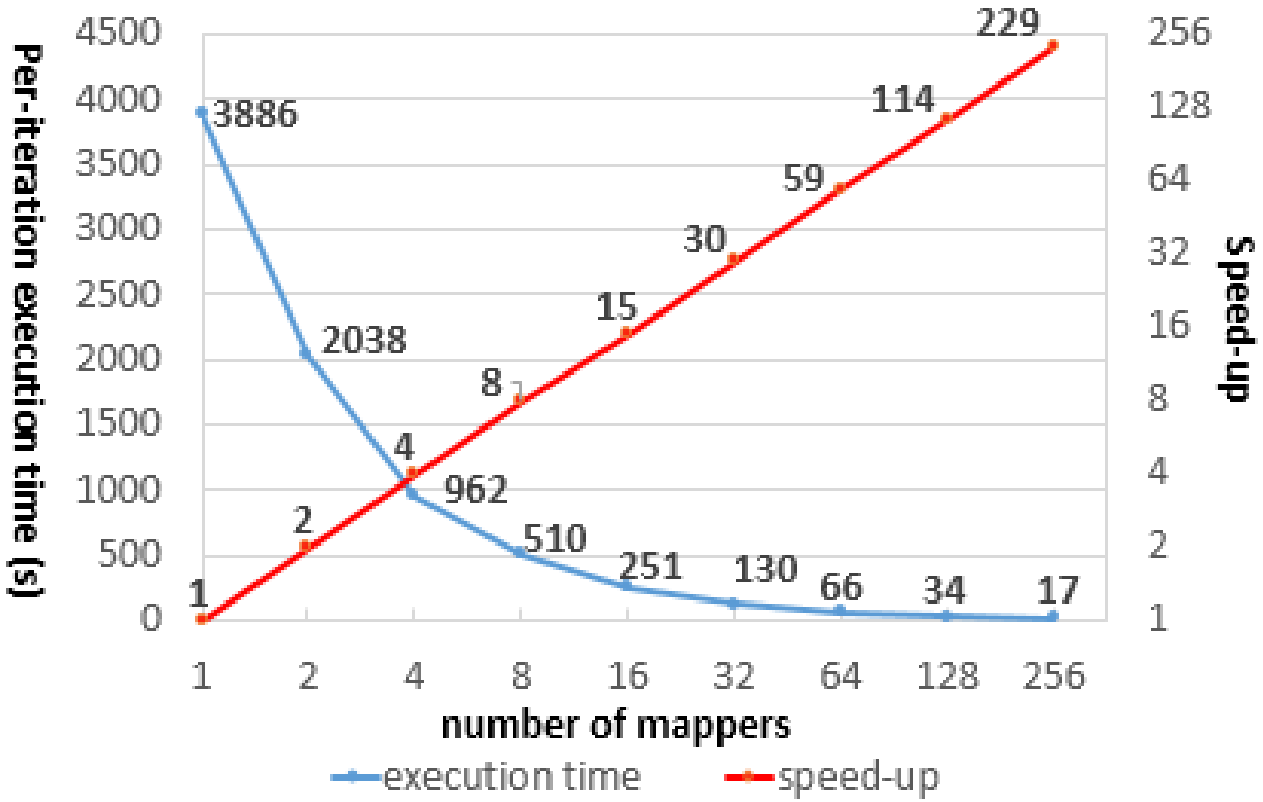


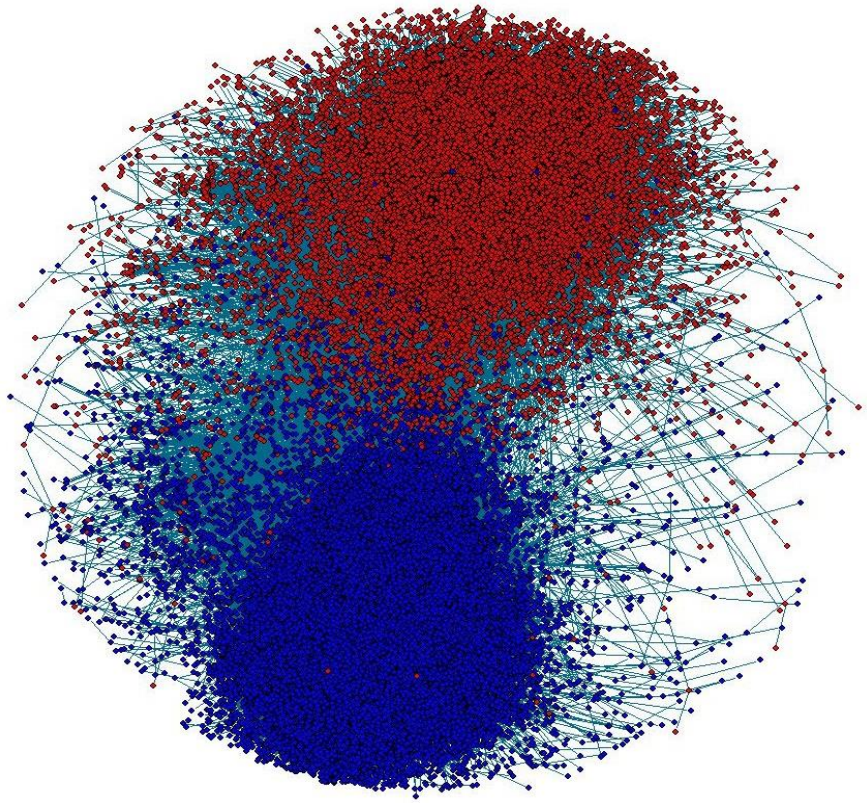
Reproduced results for 2010, extended to 2012 with a **20** times larger network

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Performance analysis

- Near linear scalability for Fruchterman-Reingold on Twister-Ivy
- Per-iteration on sequential R for 2012 network: 6035 seconds



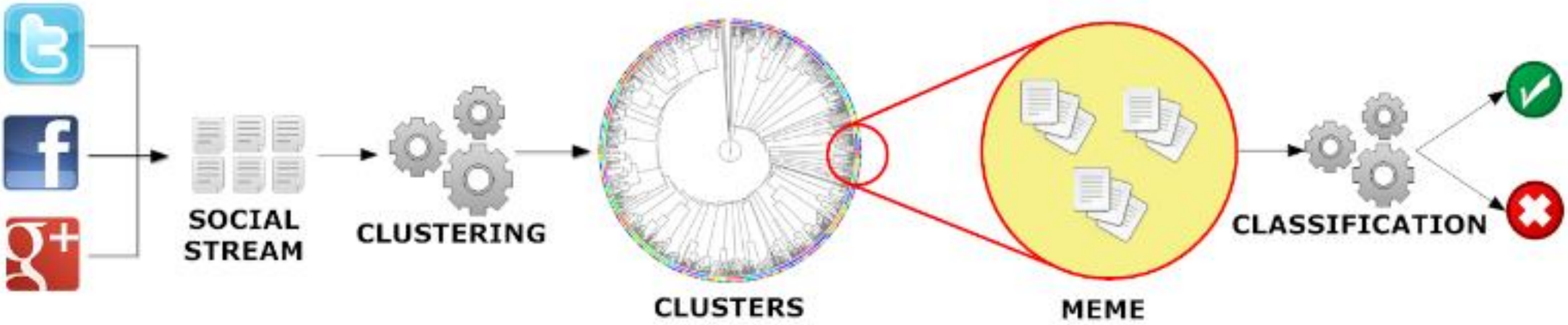


Xiaoming Gao, Judy Qiu. Social Media Data Analysis with IndexedHBase and Iterative MapReduce. MTAGS 2013
Xiaoming Gao, Judy Qiu. Supporting Queries and Analyses of Large-Scale Social Media Data with Customizable and Scalable Indexing Techniques over NoSQL Databases. CCGRID 2014.

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Parallel Tweet Online Clustering with Apache Storm

- **Analysis pipeline for meme clustering and classification :** Detecting Early Signatures of Persuasion in Information Cascades
- Implement with HBase + Hadoop (Batch) and HBase + Storm(Streaming) + ActiveMQ
- 2 million streaming tweets processed in 40 minutes; 35,000 clusters
- Storm Bolts coordinated by ActiveMQ to synchronize parallel cluster center updates



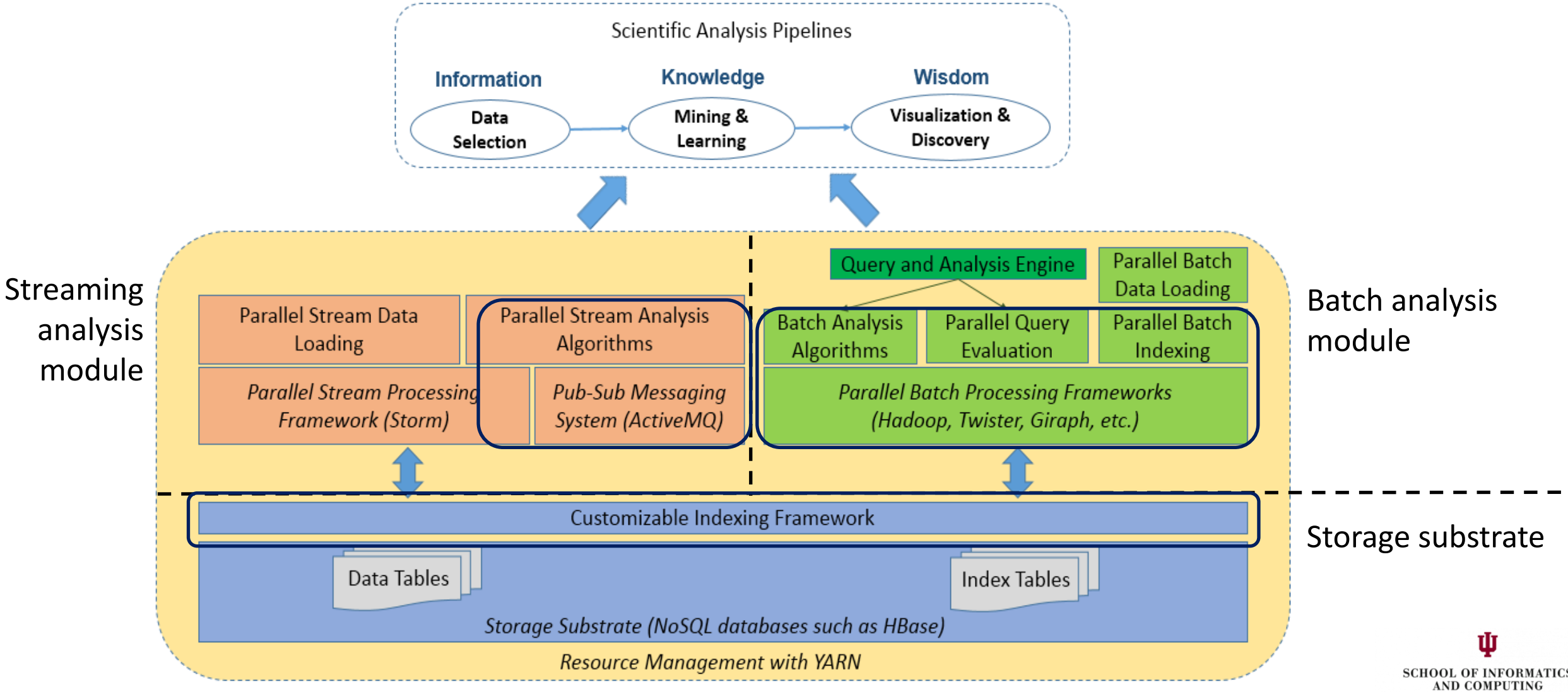
Xiaoming Gao, Emilio Ferrara, Judy Qiu, Parallel Clustering of High-Dimensional Social Media Data Streams Proceedings of CCGrid, May 4-7, 2015

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Outline

- Social media data and emerging characteristics
- Query using Pig and Hive with IndexedHBase
- Integration of query and analysis
- Summary and future work

A scalable architecture for an integrated query and analysis solution



SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Contributions and future work

- High Level Language integration (2015-2017)
  - Integrate with locality-aware workflow systems for an end-to-end solution of data parallel applications
  - Use of High level interfaces such as Pig and Hive with Harp (hadoop plugin) to support complex iterative computation and improve the flexibility of database systems through user-defined aggregations
- Streaming analysis module (2014-2015)
  - novel cluster-delta synchronization to achieve scalability
  - real-time processing of 10% Twitter stream
- Batch analysis module (2013-2014)
  - integrated analysis stack based on YARN
  - index-based analysis algorithms multiple to 10s of times faster than data scanning solutions
  - first iterative MapReduce Fruchterman-Reingold, near-linear scalability
- Storage substrate (2011-2012)
  - customizable indexing framework over NoSQL databases
  - data loading/indexing faster by multiple times
  - queries faster by one to two orders of magnitude

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

IndexedHBase

- IndexedHBase is a scalable, fault-tolerant, and indexed NoSQL table storage system.
- IndexedHBase has been successfully used to support Truthy, a public social media observatory that allows analysis of large scale (hundreds of TBs) social network data collected from Twitter's streaming API.
- **Project Link:** http://salsaproj.indiana.edu/IndexedHBase/
- **Source Code Link:** https://github.com/Truthy-team/IndexedHBase

SCHOOL OF INFORMATICS
AND COMPUTING
INDIANA UNIVERSITY
Bloomington

The Harp Library

- Harp is an implementation designed in a pluggable way to bring high performance to the Apache Big Data Stack and bridge the differences between Hadoop ecosystem and HPC system through a clear communication abstraction, which did not exist before in the Hadoop ecosystem.
- Hadoop Plugin that targets Hadoop 2.2.0
- Provides implementation of the collective communication abstractions and MapCollective programming model
- **Project Link:** http://salsaproj.indiana.edu/harp/index.html
- **Source Code Link:** https://github.com/jessezbj/harp-project

We built Map-Collective as a unified model to improve the performance and expressiveness of Big Data tools. We ran Harp on K-means, Graph Layout, and Multidimensional Scaling algorithms with realistic application datasets over 4096 cores on the IU BigRed II Supercomputer (Cray/Gemini) where we have achieved linear speedup.

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington