Portable Data Mining on Azure and HPC Platform

Judy Qiu, Thilina Gunarathne, Bingjing Zhang, Xiaoming Gao, Fei Teng

SALSA HPC Group

http://salsahpc.indiana.edu

School of Informatics and Computing

Indiana University

Data Intensive Iterative Applications

- Growing class of applications
  - Clustering, data mining, machine learning & dimension reduction applications Expectation Maximization
  - Driven by data deluge & emerging computation fields
  - Lots of scientific applications

$k \leftarrow 0;$

MAX $\leftarrow$ maximum iterations

$\delta^{[0]} \leftarrow$ initial delta value

while ($k <$ MAX_ITER || $f(\delta^{[k]}, \delta^{[k-1]})$)

    foreach datum in data

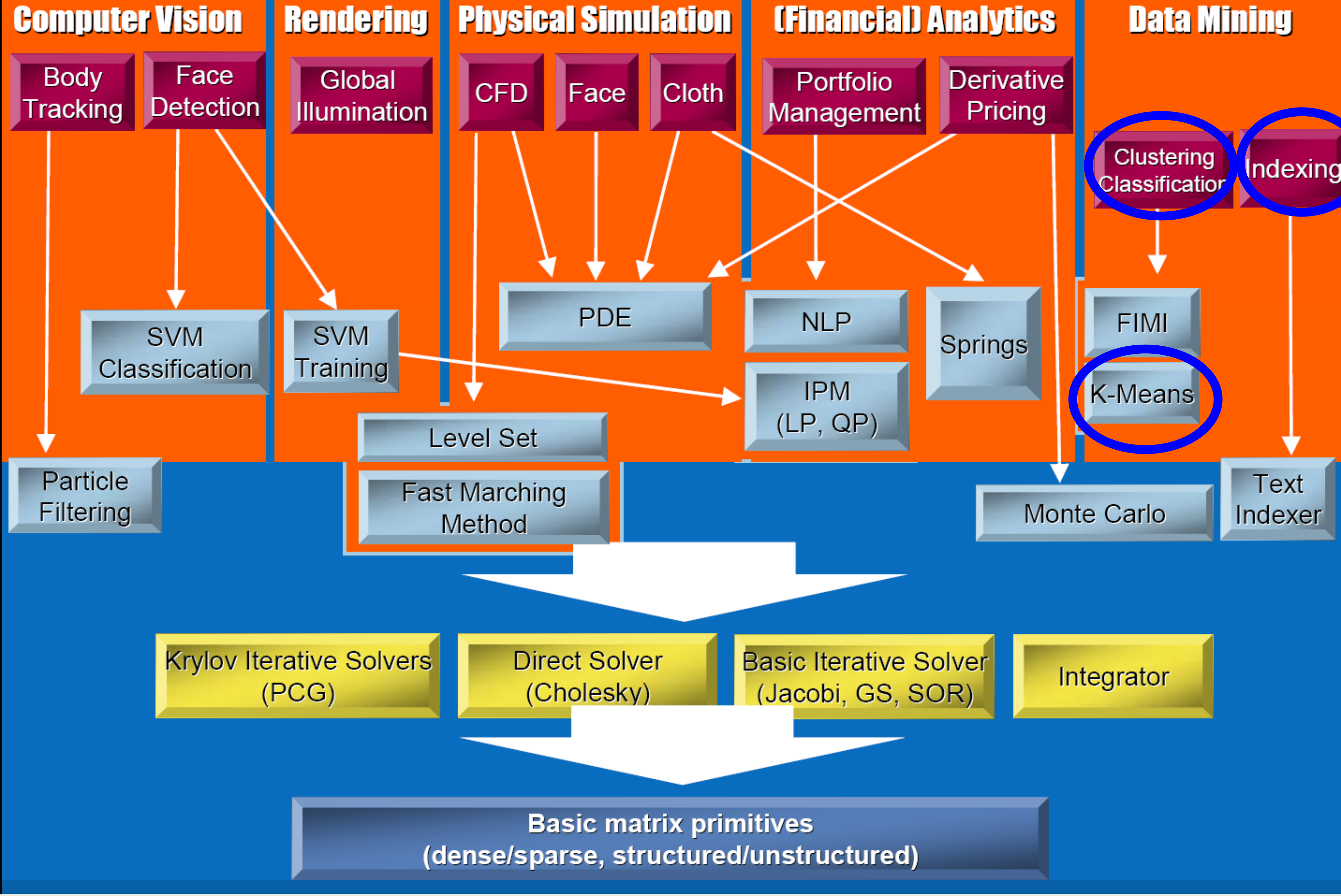
        $\beta[\text{datum}] \leftarrow \text{process}(\text{datum}, \delta^{[k]})$

    end foreach

    $\delta^{[k+1]} \leftarrow \text{combine}(\beta[\,])$

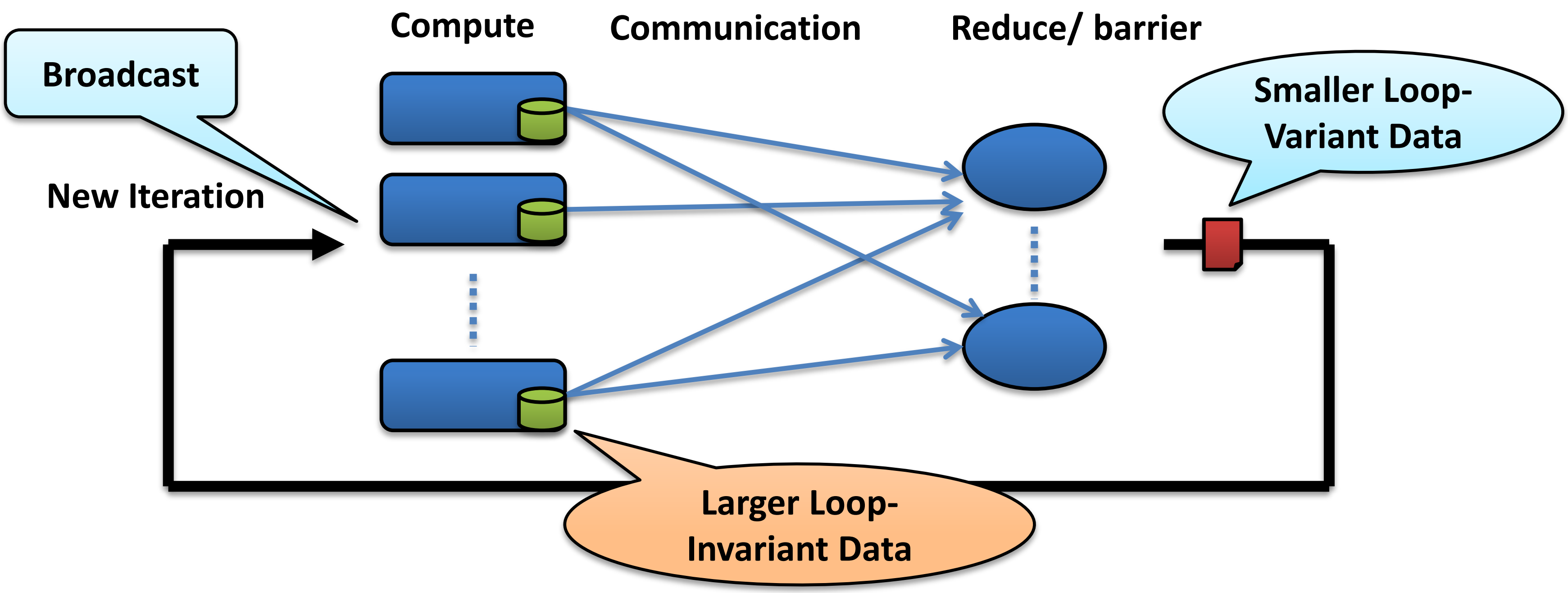
    $k \leftarrow k+1$

end while



Intel's Application Stack

Data Intensive Iterative Applications



- Common Characteristics
- Compute (map) followed by LARGE communication Collectives (reduce)

Iterative MapReduce

- MapReduceMerge

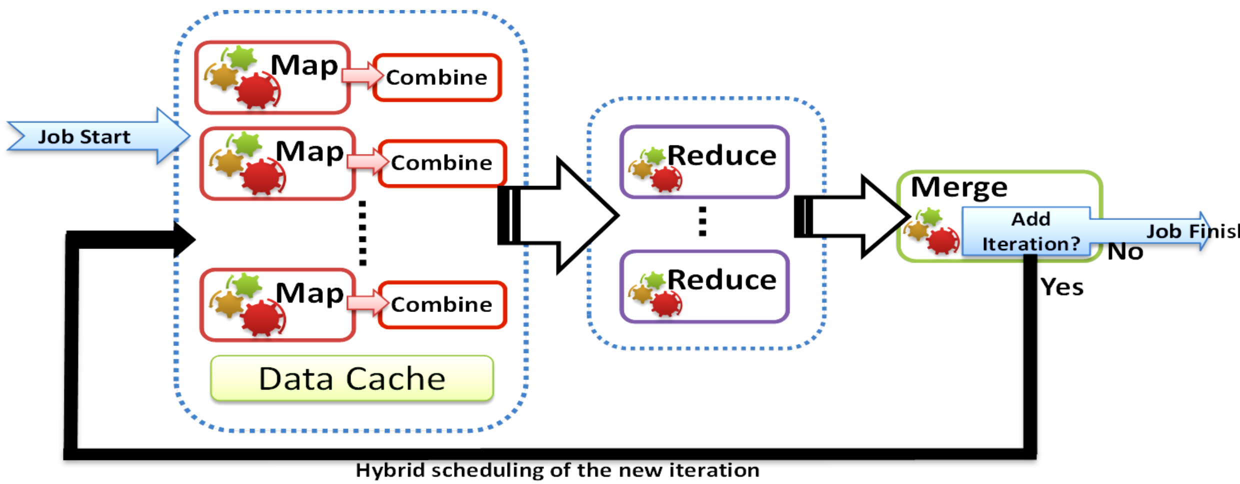
Map → Combine → Shuffle → Sort → Reduce → Merge → Broadcast

- Extensions to support additional broadcast (+other) input data

Map(<key>, <value>, list_of <key,value>)

Reduce(<key>, list_of <value>, list_of <key,value>)

Merge(list_of <key,list_of<value>>,list_of <key,value>)



Parallel Data Analysis using Twister

- **Data mining and Data analysis Applications**
  - *Next Generation Sequencing*
  - *Image processing*
  - *Search Engine*
  - ….
- **Algorithms**
  - *Multidimensional Scaling (MDS)*
  - *Clustering (K-means)*
  - *Indexing*
  - ….

Challenges

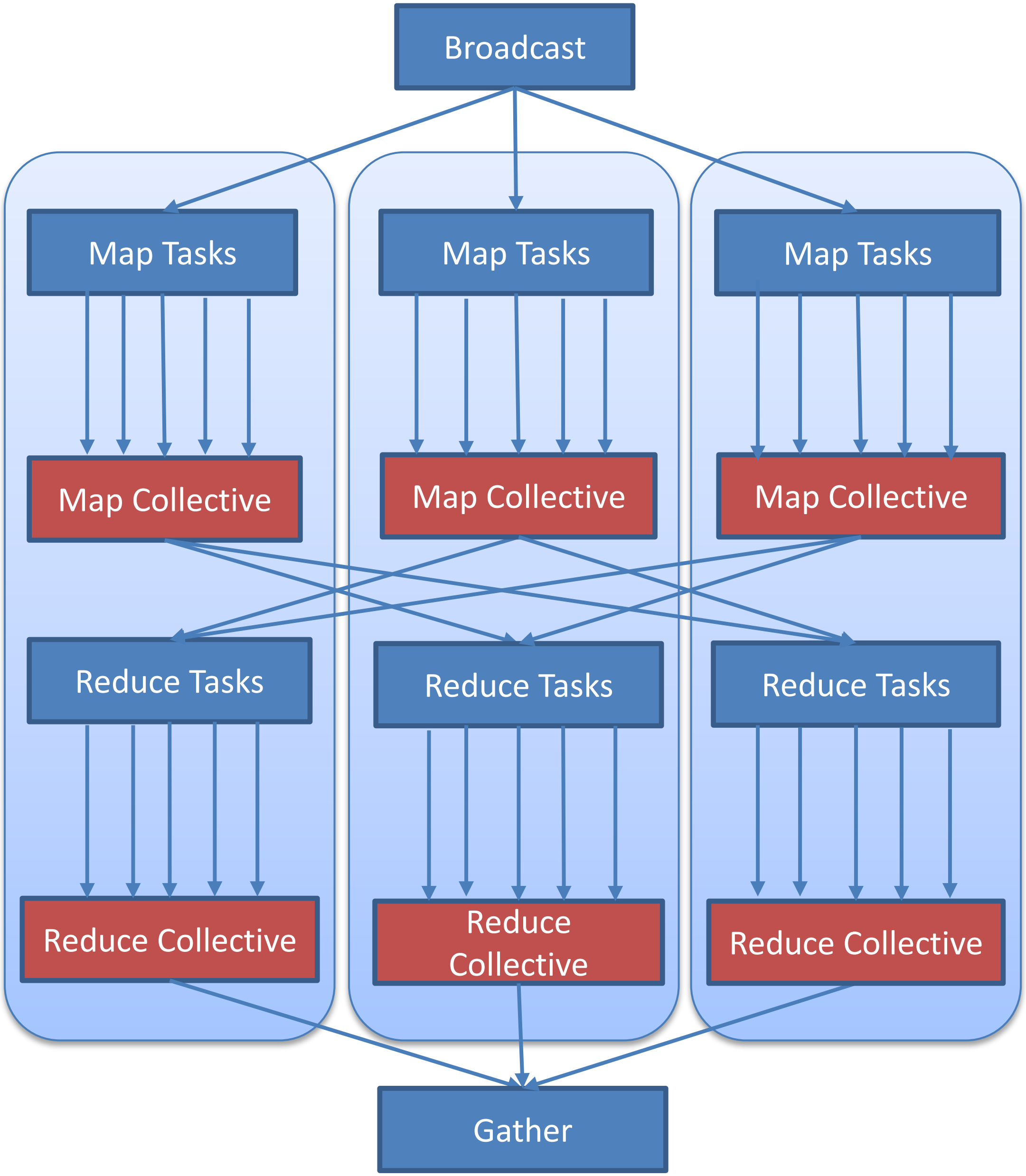
- **Traditional MapReduce and classical parallel runtimes cannot solve iterative algorithms efficiently**
  - *Hadoop: Repeated data access to HDFS, no optimization to data caching and data transfers*
  - *MPI: no natural support of fault tolerance and programming interface is complicated*
- **Interoperability**

Current and Future Work

- **Collective Communication**
- **Fault tolerance**
- **Distributed Storage**
- **High level language**

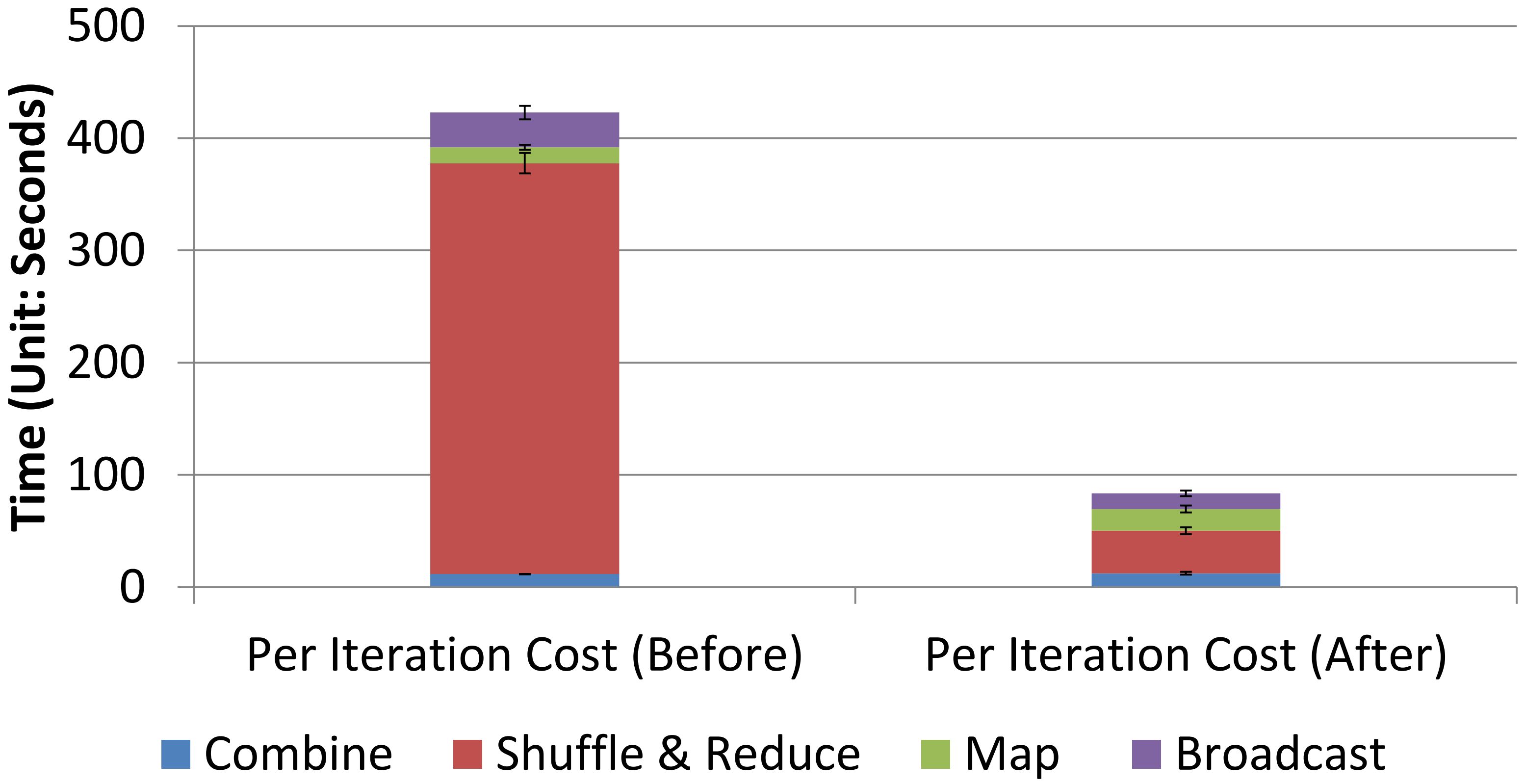
Twister Collective Communications

- Broadcasting
  - Data could be large
  - Chain & MST
- Map Collectives
  - Local merge
- Reduce Collectives
  - Collect but no merge
- Combine
  - Direct download or Gather

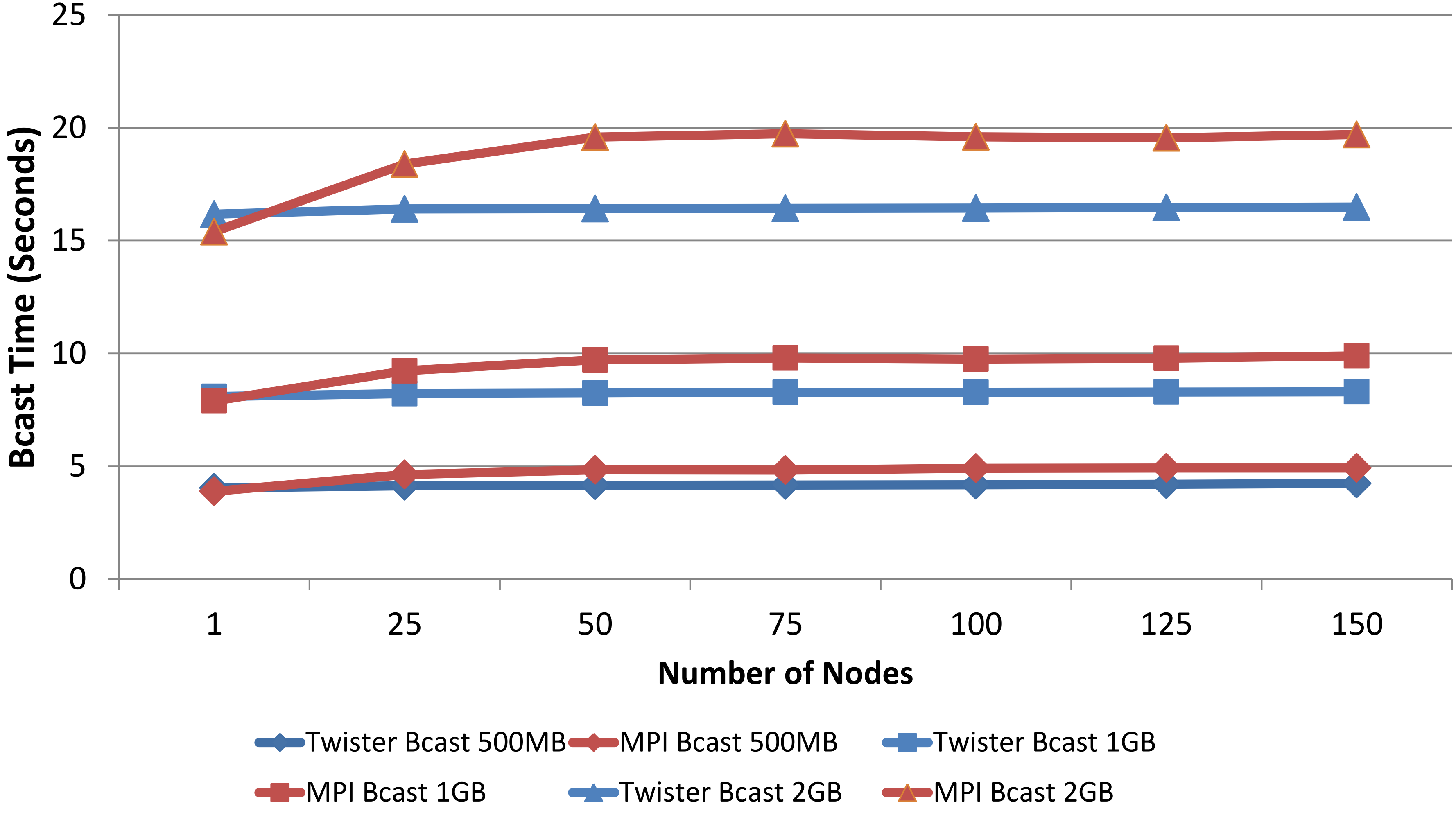


Twister Broadcast Comparison

One-to-All vs. All-to-All implementations

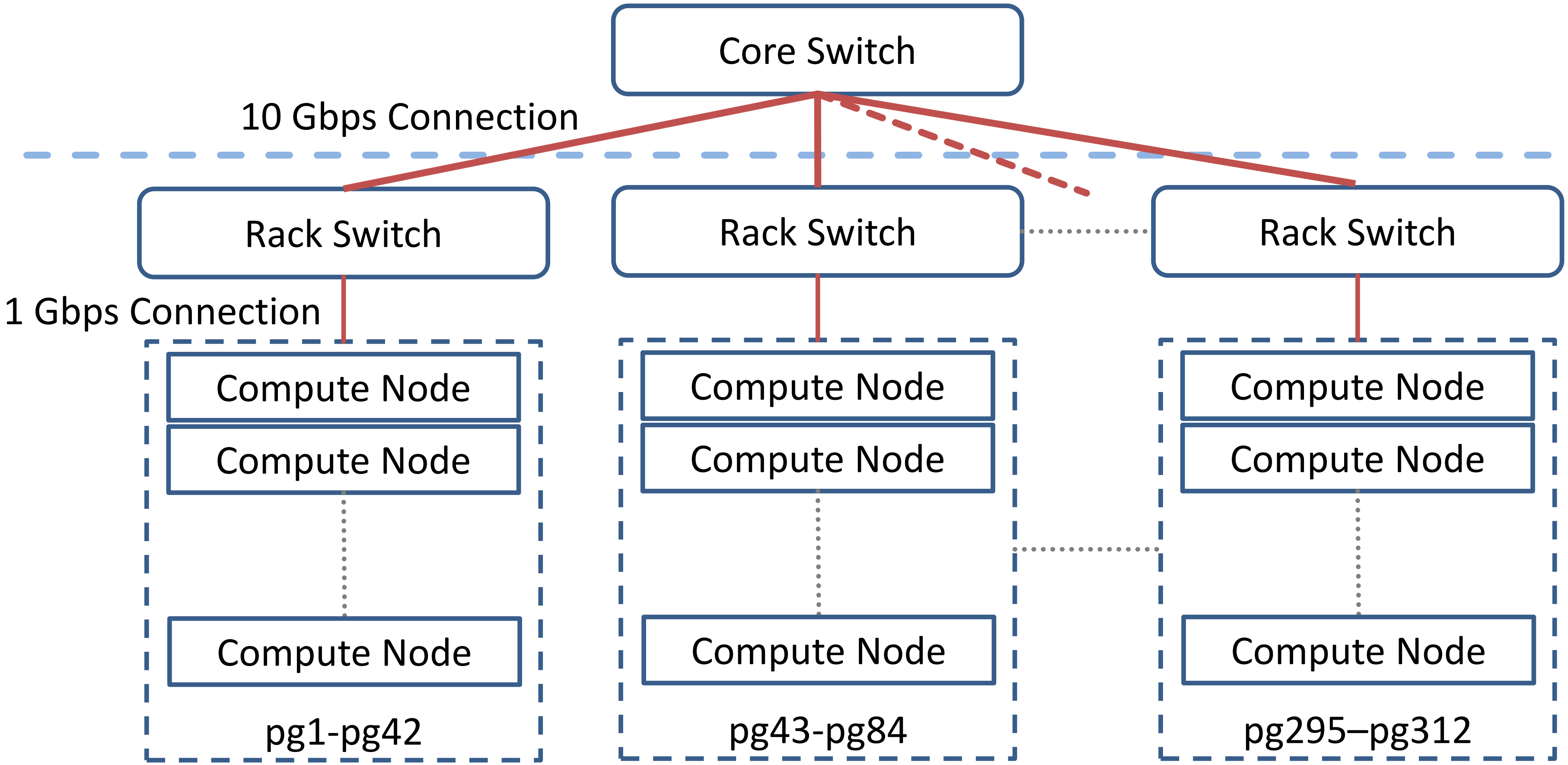


Bcast Byte Array on PolarGrid (Fat-Tree Topology with 1Gbps Ethernet): Twister v. MPI (OpenMPI)

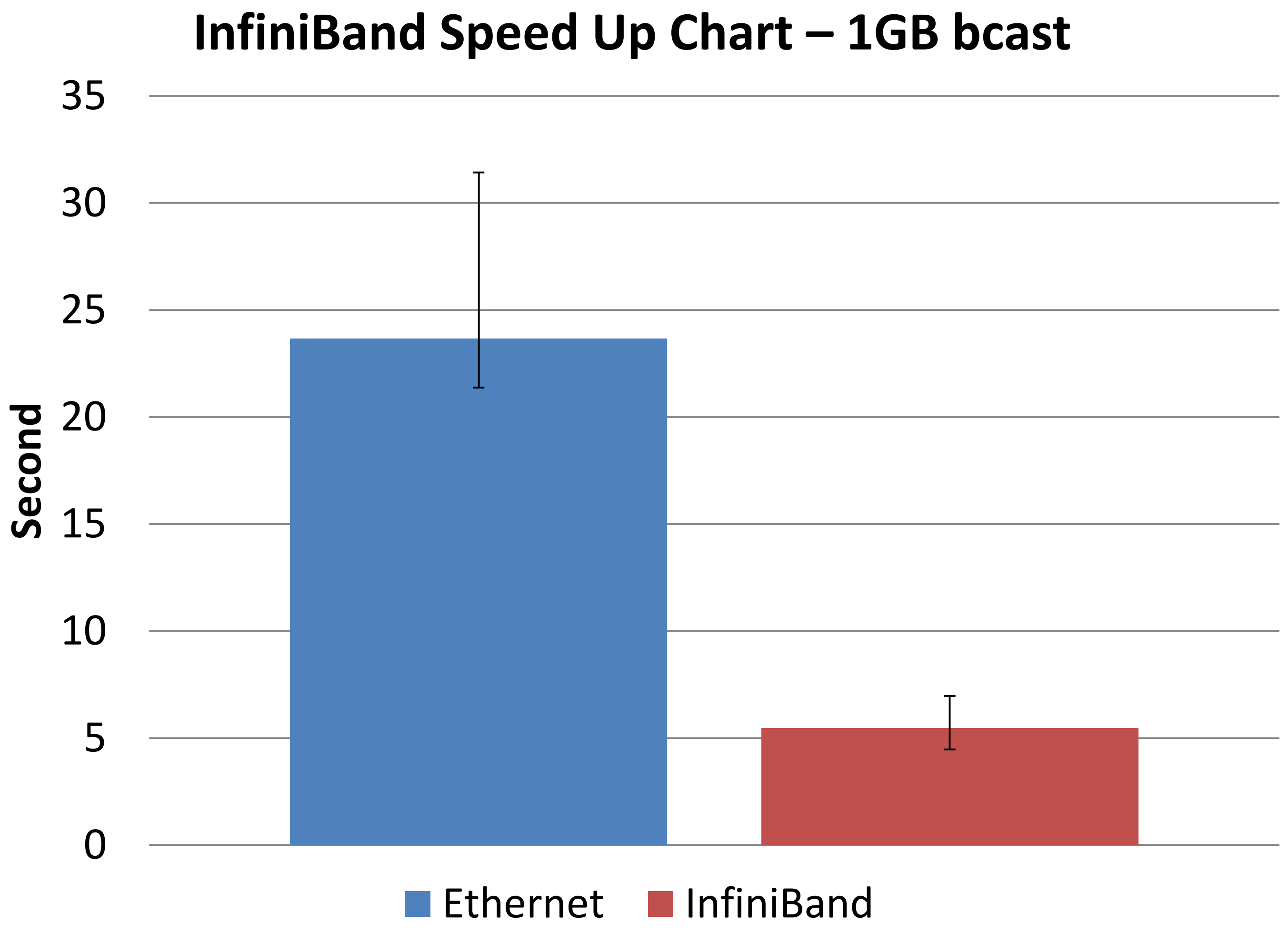


We are optimizing Collectives needed in data mining

Collective algorithm uses Topology-aware Pipeline

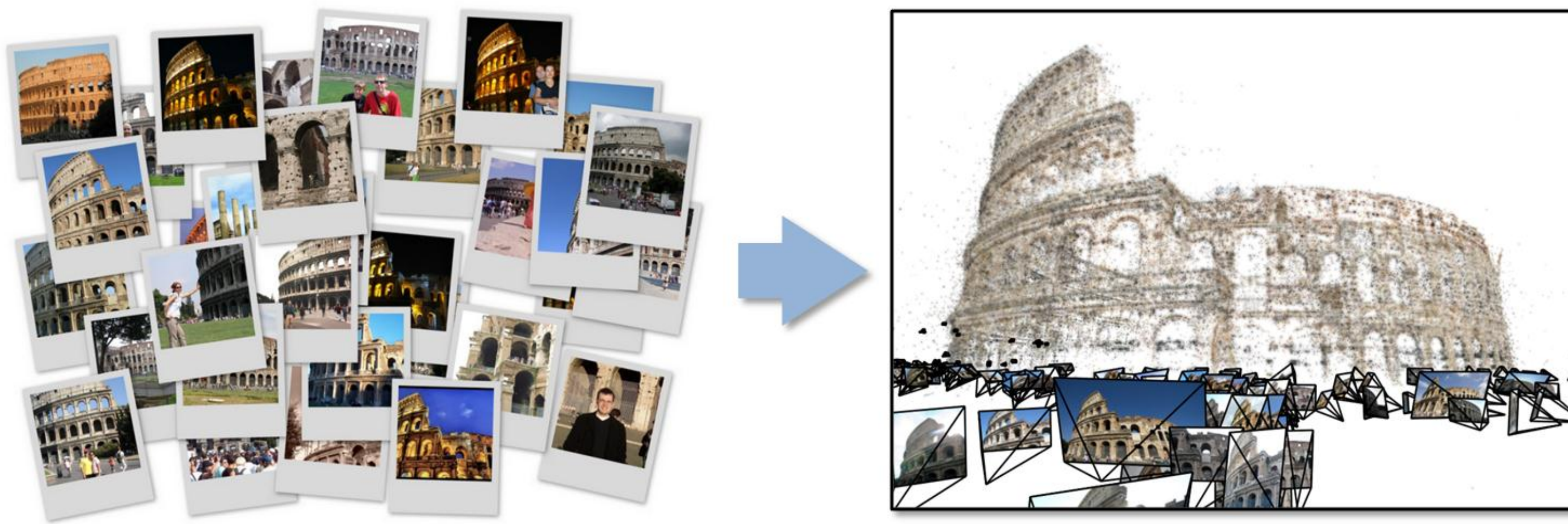


Twister Broadcast Comparison:
Ethernet vs. InfiniBand (Oak Ridge)



Data Intensive Kmeans Clustering

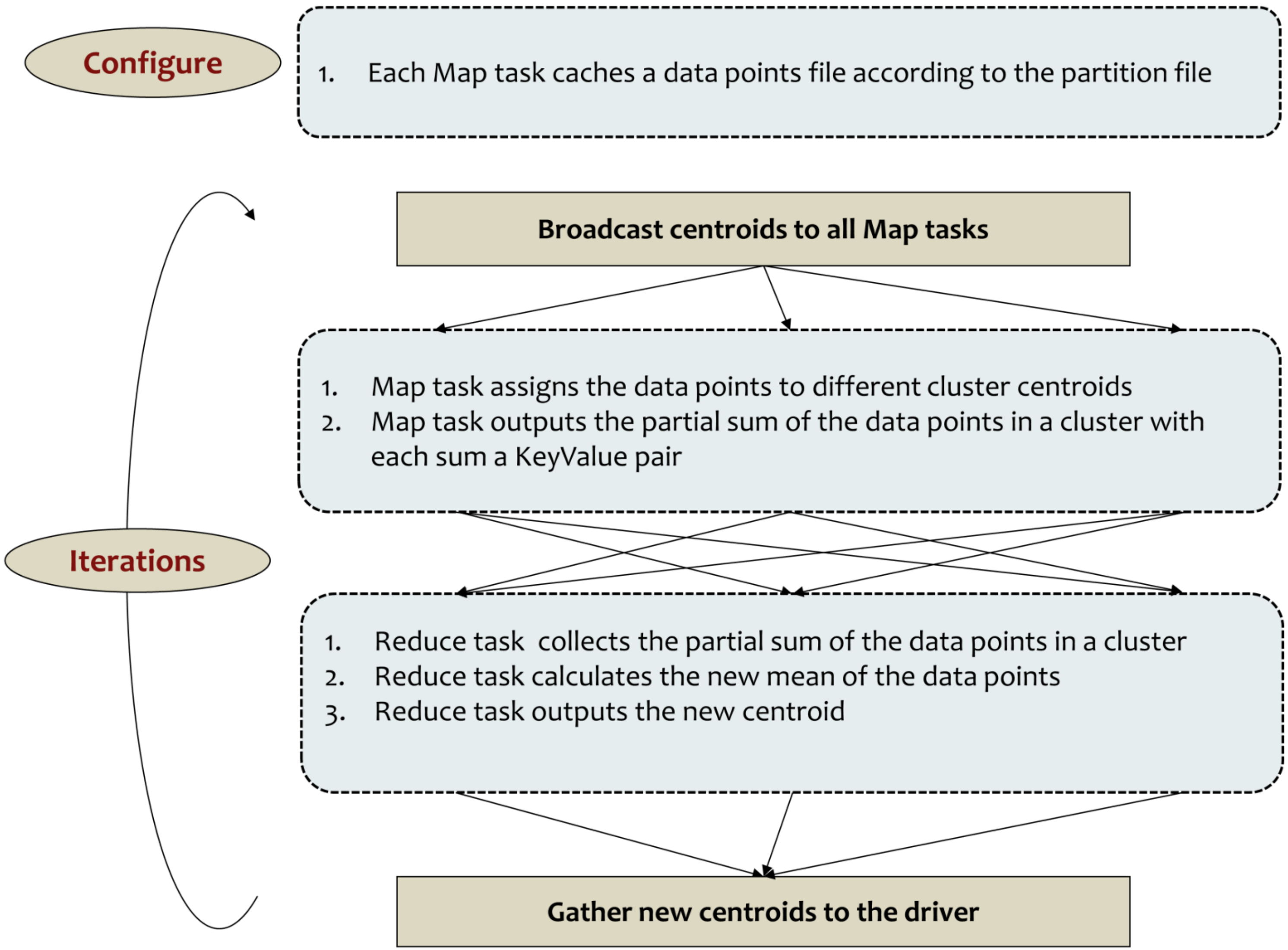
– *Image Classification:* ***1.5 TB;*** 500 features per image;10k clusters
1000 Map tasks; 1GB data transfer per Map task node



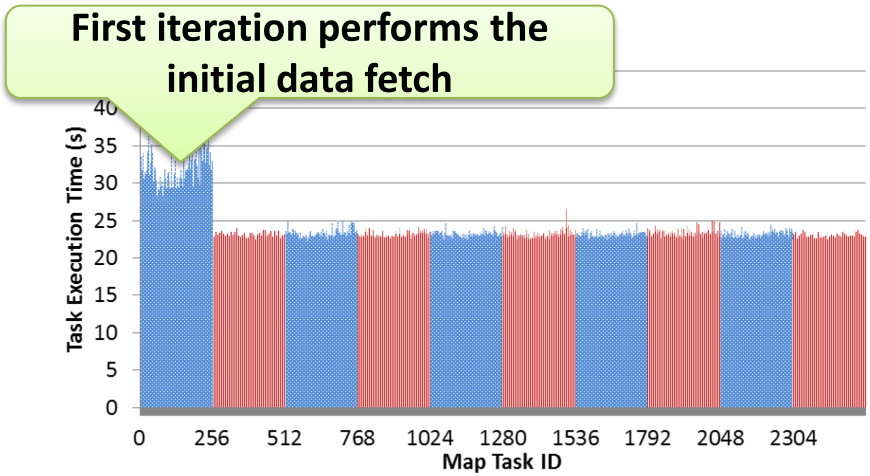
High Dimensional Data

- K-means Clustering algorithm is used to cluster the images with similar features.
- In image clustering application, each image is characterized as a data point with 512 dimensions. Each value ranges from 0 to 255.
- Currently, we are able able to process 10 million images with 166 machines and cluster the vectors to 1 million clusters
  - Need 180 million images
- Improving algorithm (Elkan) and runtime (Twister Collectives)

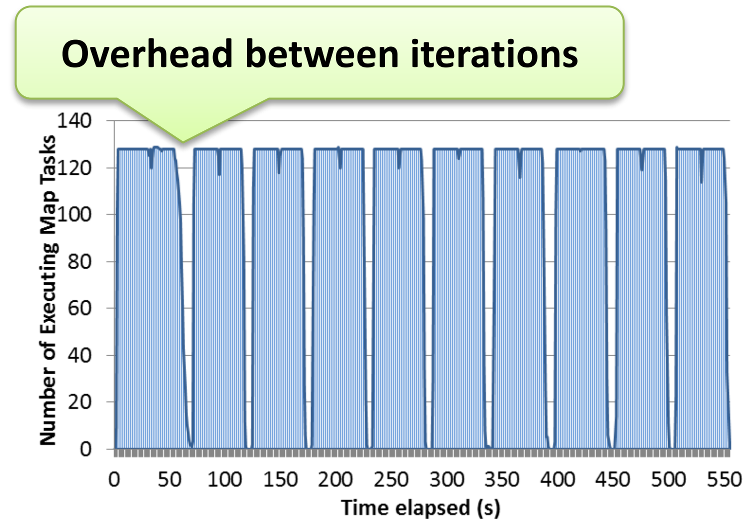
Twister Kmeans Clustering



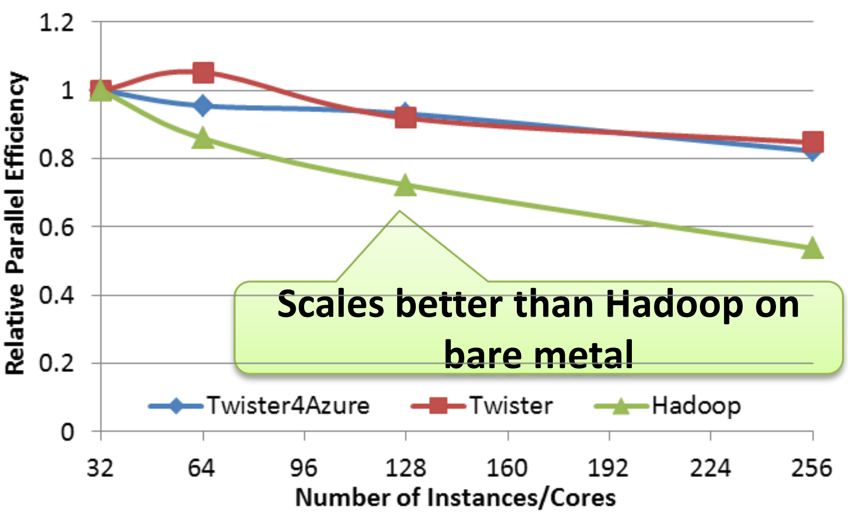
Kmeans Clustering on Twister4Azure



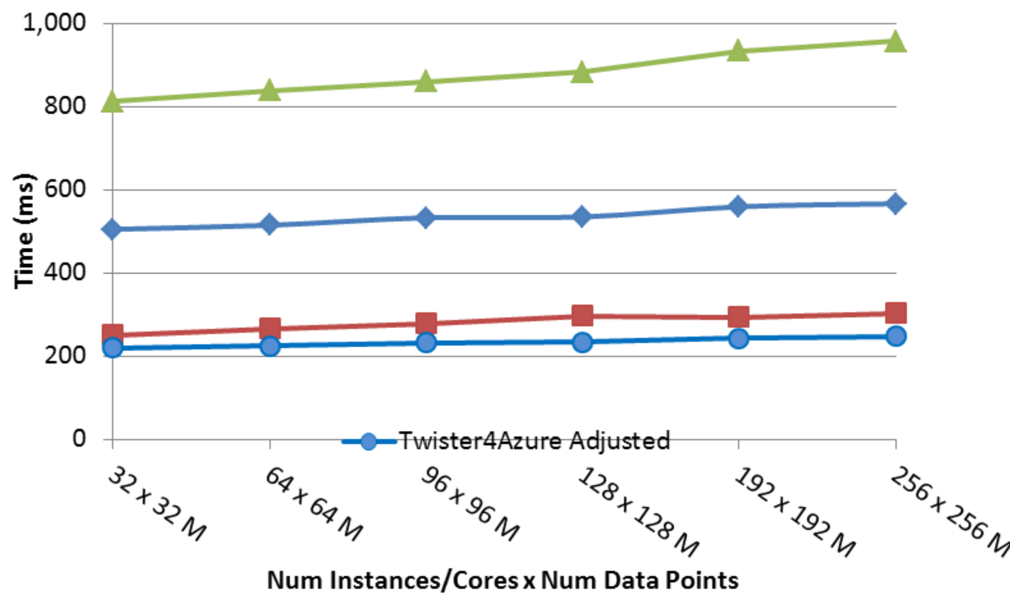
Task Execution Time Histogram



Number of Executing Map Task Histogram



Strong Scaling with 128M Data Points

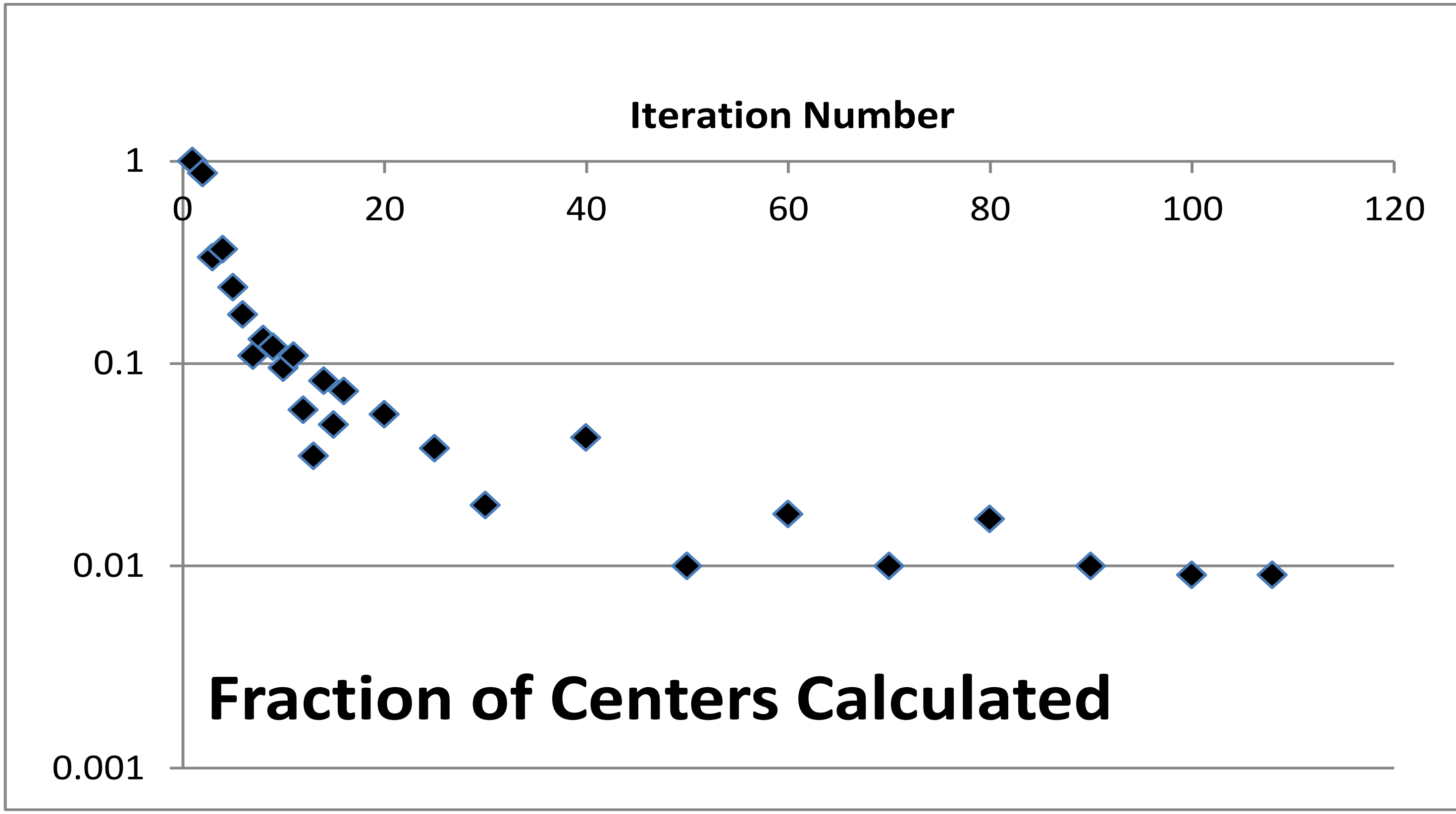


Weak Scaling

Triangle Inequality and Kmeans

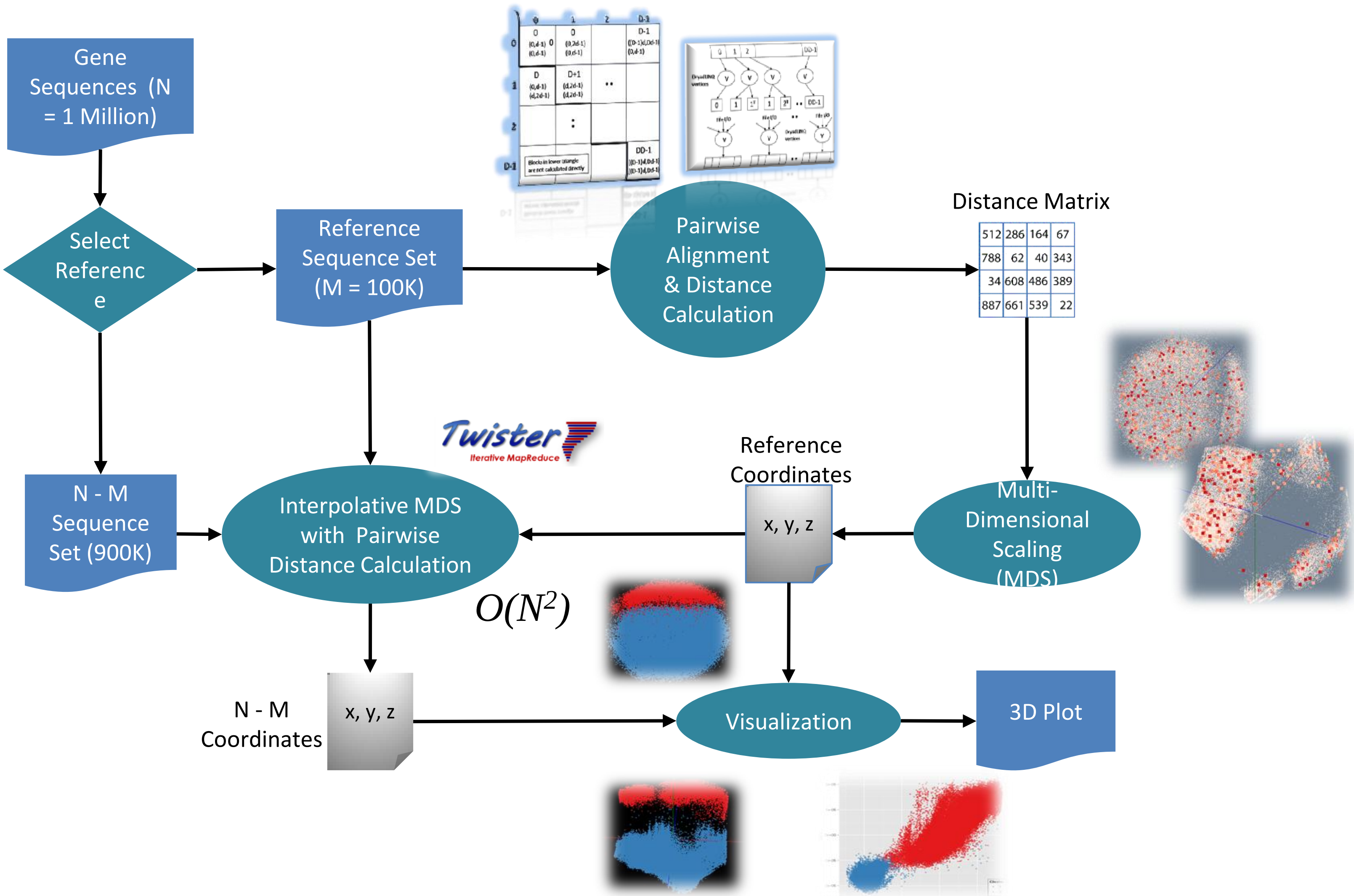
- Dominant part of Kmeans algorithm is finding nearest center to each point
  O(#Points * #Clusters * Vector Dimension)
- Simple algorithms finds
  min over centers c: d(x, c) = distance(point x, center c)
- But most of d(x, c) calculations are wasted as much larger than minimum value
- Elkan (2003) showed how to use triangle inequality to speed up using relations like
  - d(x, c2) >= d(x,c2-last) – d(c2, c2-last) and
  - d(x, c2) >= d(c1, c2) – d(x,c1)
  - c2-last position of center at last iteration; c1 c2 two centers
- So compare estimate of d(x, c2) with d(x, c1) where c1 is nearest cluster at last iteration
- Complexity reduced by a factor = Vector Dimension and so this important in clustering high dimension spaces such as social imagery with 500 or more features per image

Early Results on Elkan's Algorithm



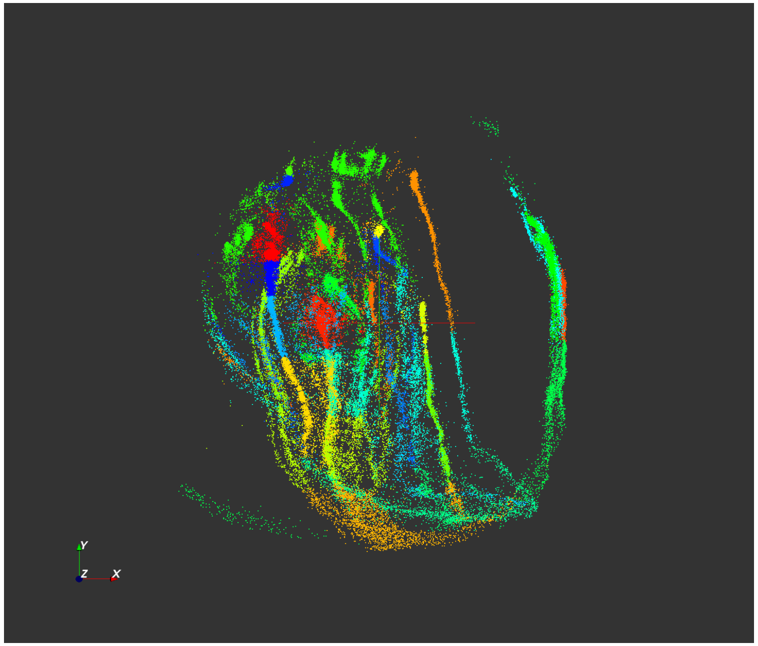
- Graph shows fraction of distances d(x, c) that need to be calculated each iteration for a test data set
- Only 5% on average of distance calculations needed
- 200K points, 124 centers, Vector Dimension 74

Bioinformatics Pipeline

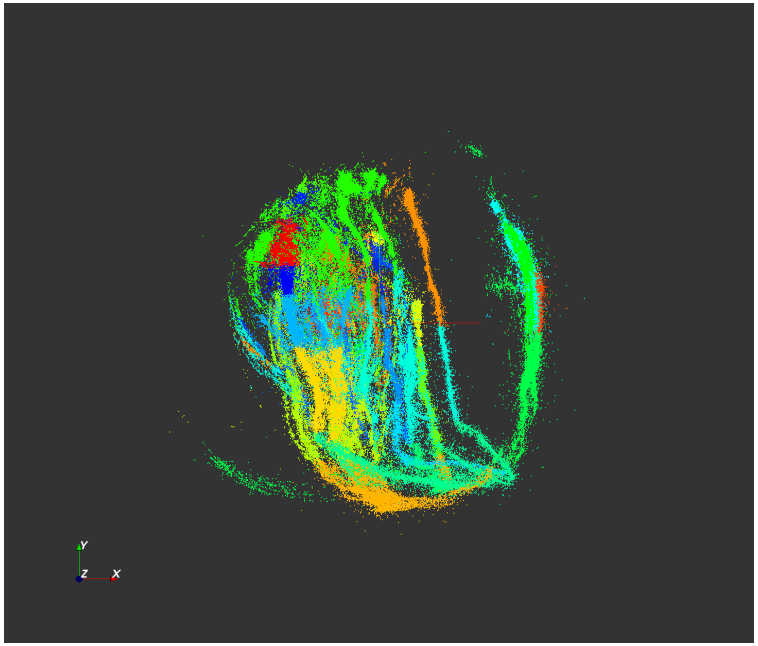


Million Sequence Challenge

- Input DataSize: 680k
- Sample Data Size: 100k
- Out-Sample Data Size: 580k
- Test Environment: PolarGrid with 100 nodes, 800 workers.

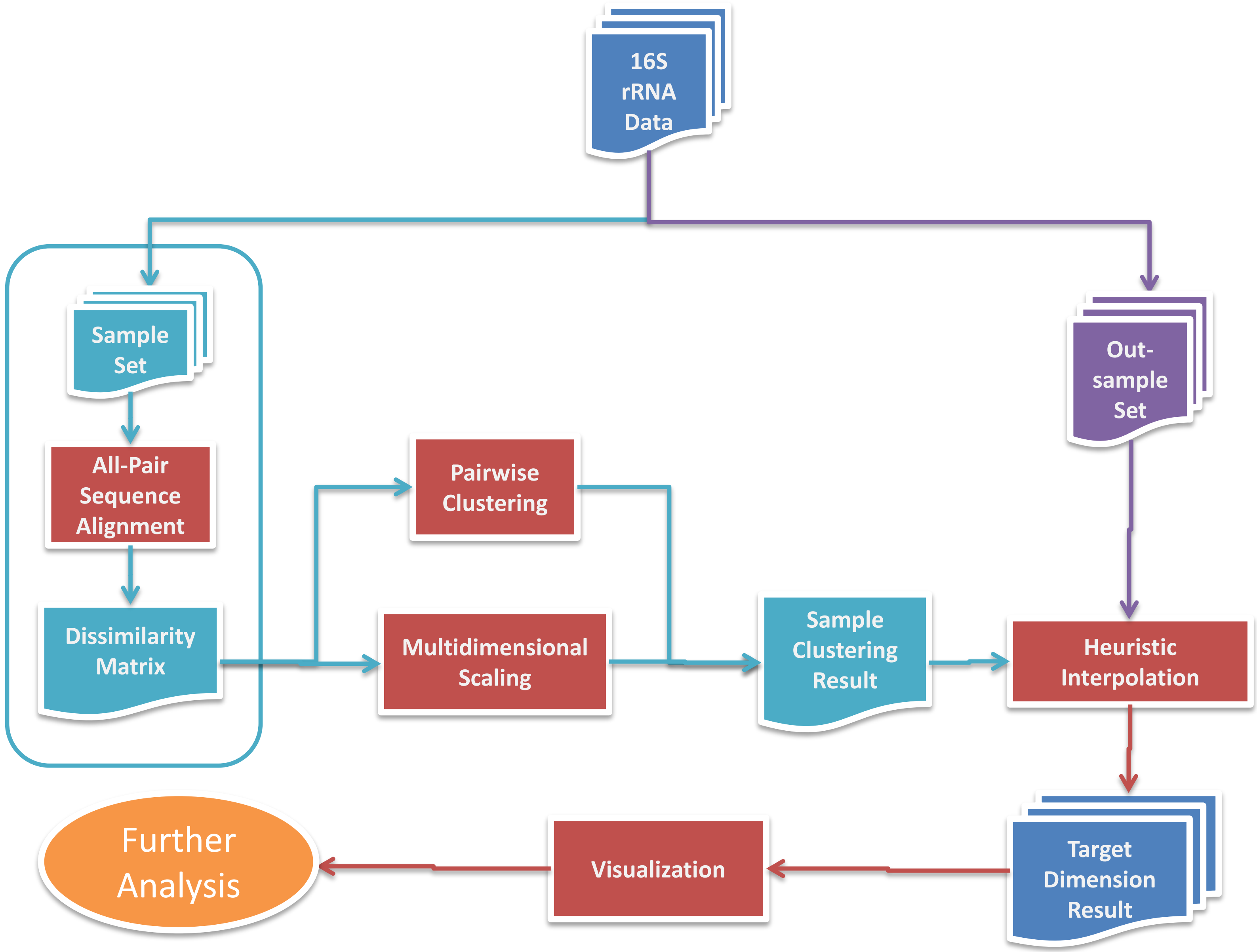


100k sample data



680k data

DACIDR (A Deterministic Annealing Clustering and Interpolative Dimension Reduction Method) Flow Chart



Dimension Reduction Algorithms

- **Multidimensional Scaling (MDS) [1]**
  - Given the proximity information among points.
  - Optimization problem to find mapping in target dimension of the given data based on pairwise proximity information while minimize the objective function.
  - Objective functions: STRESS (1) or SSTRESS (2)

$$\sigma(\boldsymbol{X}) = \sum_{i<j\leq N} w_{ij}(d_{ij}(\boldsymbol{X}) - \delta_{ij})^2 \quad (1)$$

$$\sigma^2(\boldsymbol{X}) = \sum_{i<j\leq N} w_{ij}[(d_{ij}(\boldsymbol{X}))^2 - (\delta_{ij})^2]^2 \quad (2)$$

  - Only needs pairwise distances δ_{ij} between original points (typically not Euclidean)
  - $d_{ij}(\mathbf{X})$ is Euclidean distance between mapped (3D) points

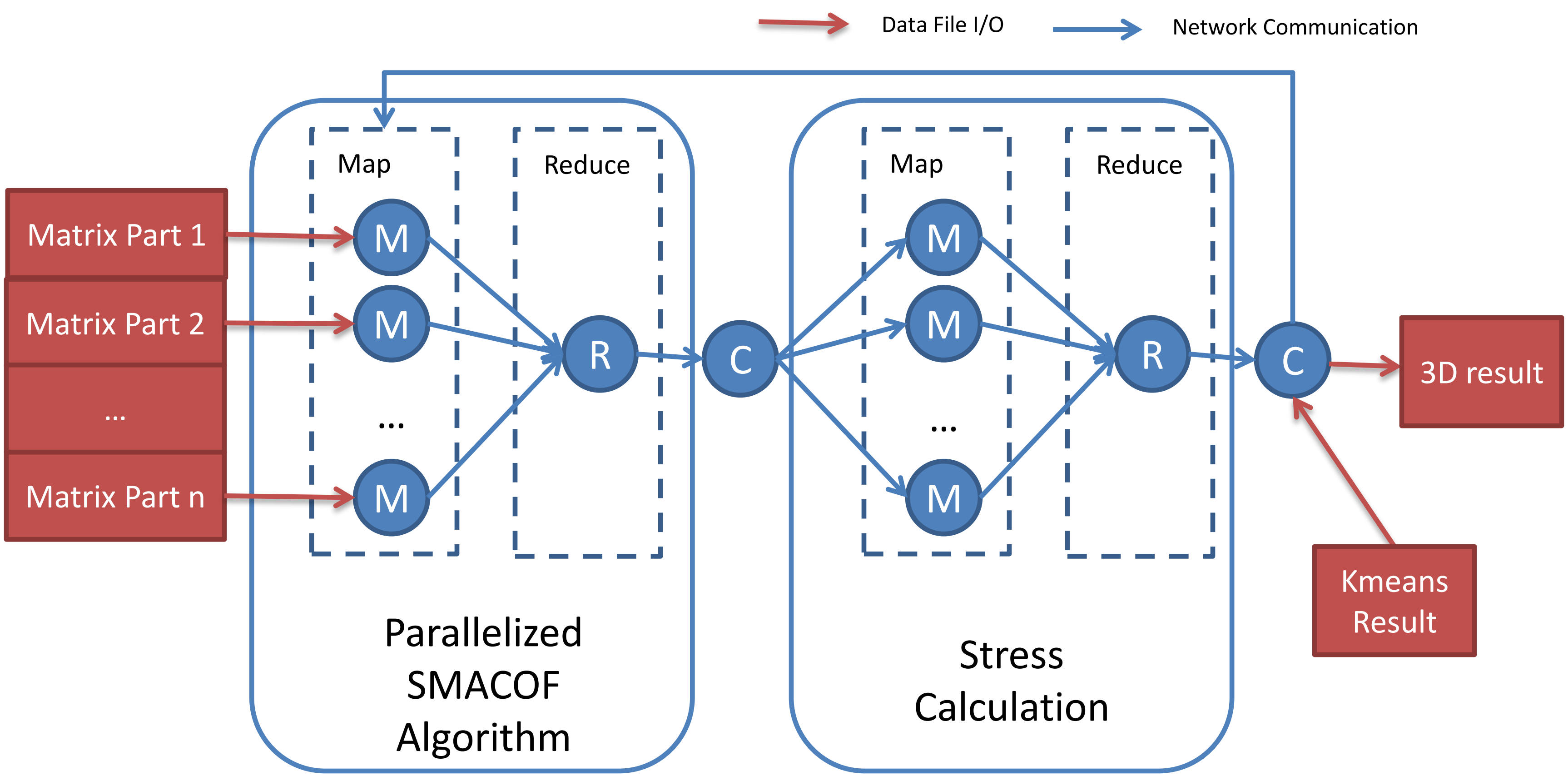
- **Generative Topographic Mapping (GTM) [2]**
  - Find optimal K-representations for the given data (in 3D), known as K-cluster problem (NP-hard)
  - Original algorithm use EM method for optimization
  - Deterministic Annealing algorithm can be used for finding a global solution
  - Objective functions is to maximize log-likelihood

$$\mathcal{L}(\boldsymbol{W}, \beta) = \sum_{j=1}^{N} \ln \left\{ \frac{1}{K} \sum_{i=1}^{K} \mathcal{N}(\boldsymbol{x}_j | f(\boldsymbol{z}_i; \boldsymbol{W}), \beta) \right\}$$

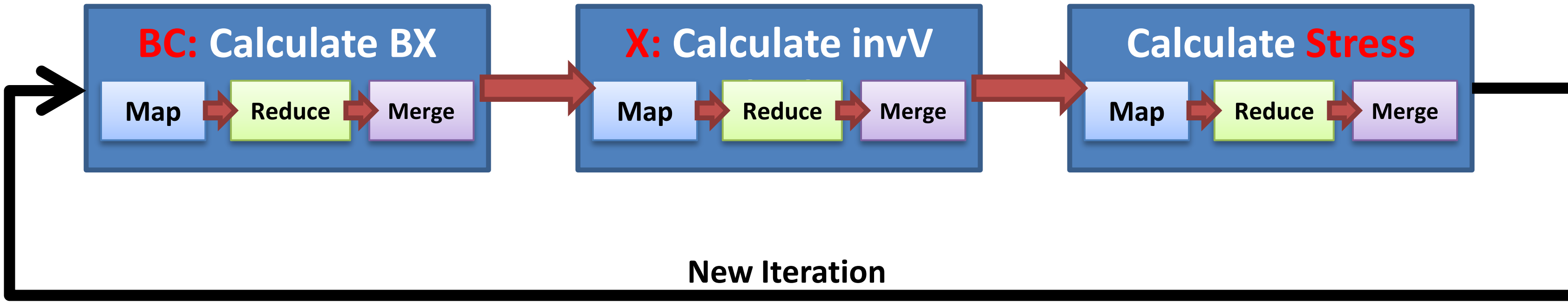
[1] I. Borg and P. J. Groenen. *Modern Multidimensional Scaling: Theory and Applications. Springer, New* York, NY, U.S.A., 2005.
[2] C. Bishop, M. Svens´en, and C. Williams. GTM: The generative topographic mapping. *Neural computation,* 10(1):215–234, 1998.

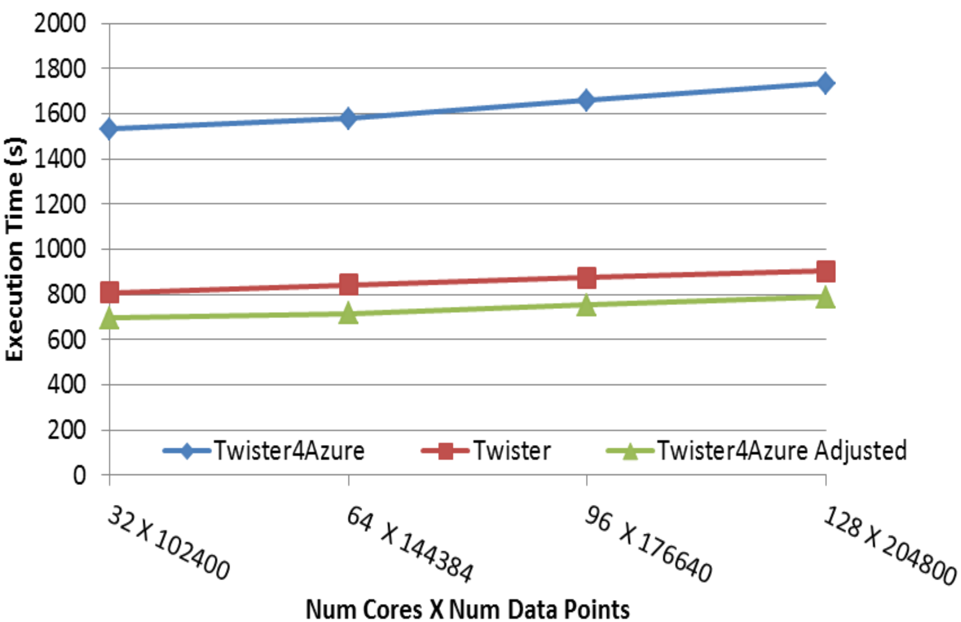
Multidimensional Scaling

- Scaling by Majorizing a Complicated Function
- Can be merged to Kmeans result

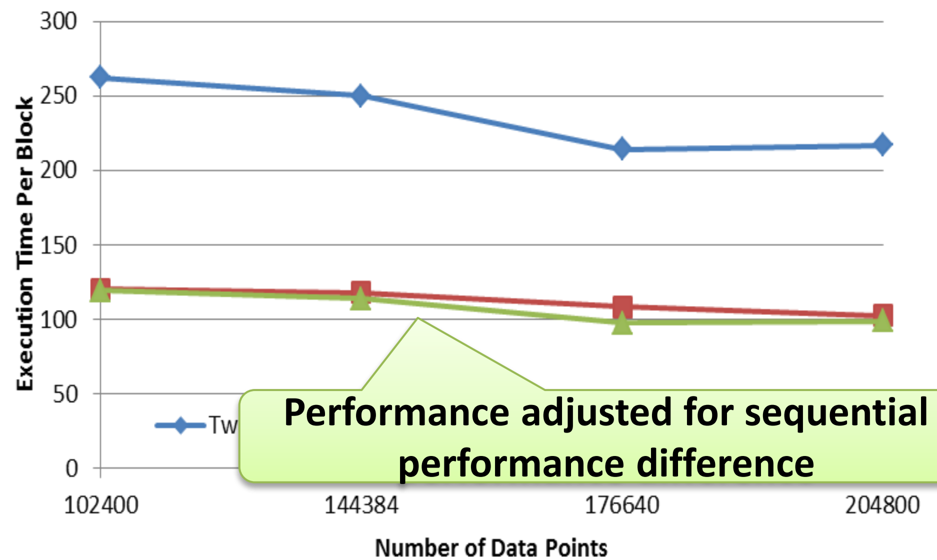


Multi Dimensional Scaling on Twister (Linux), Twister4Azure and Hadoop





Weak Scaling

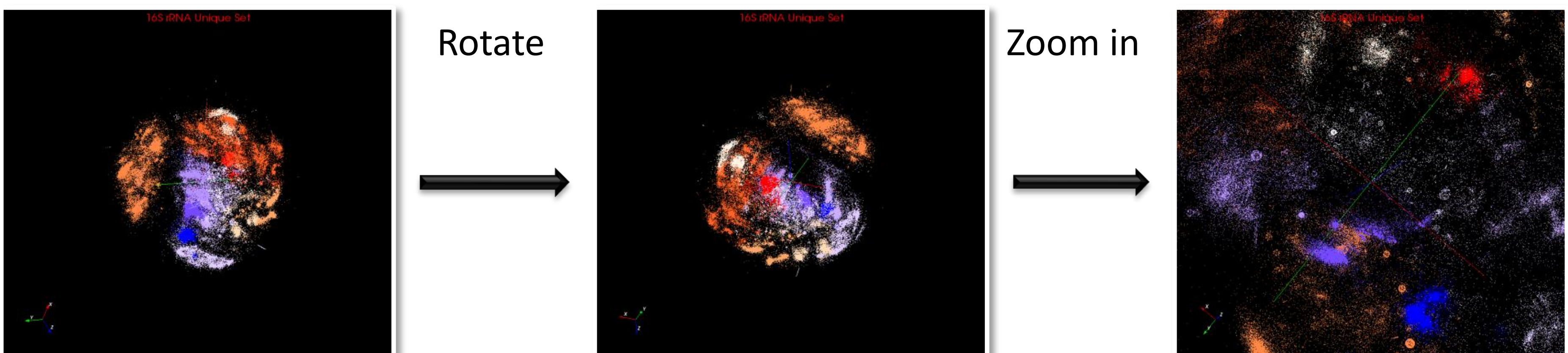


Data Size Scaling

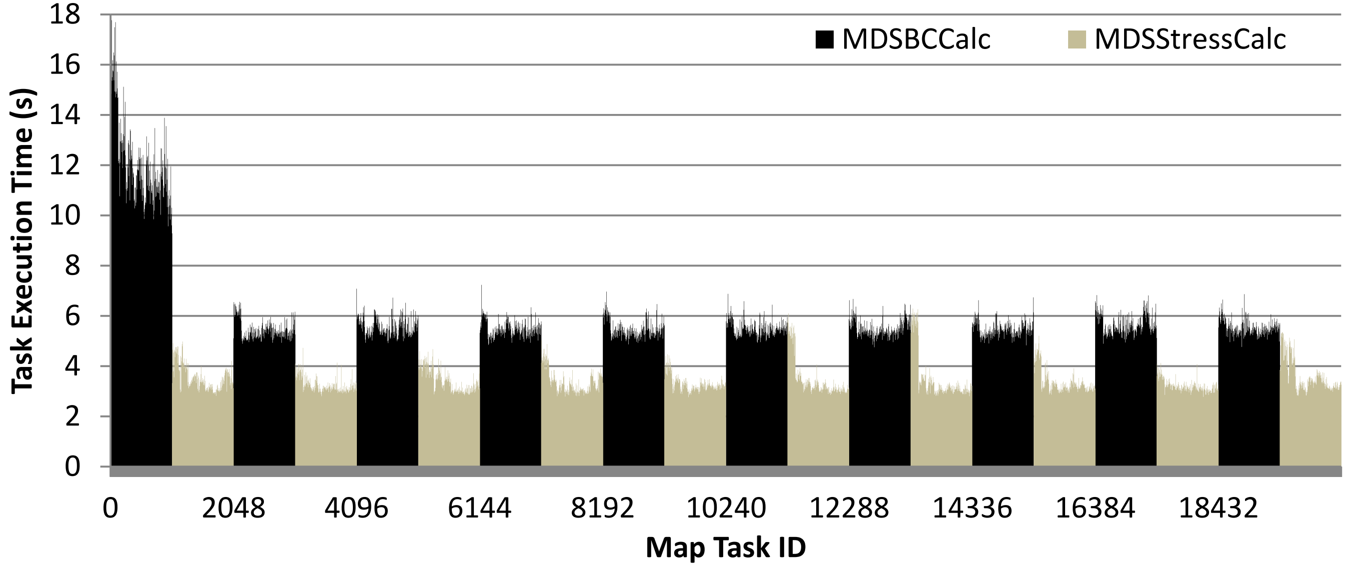
Scalable Parallel Scientific Computing Using Twister4Azure. Thilina Gunarathne, BingJing Zang, Tak-Lon Wu and Judy Qiu. Submitted to Journal of Future Generation Computer Systems. (Invited as one of the best 6 papers of UCC 2011)

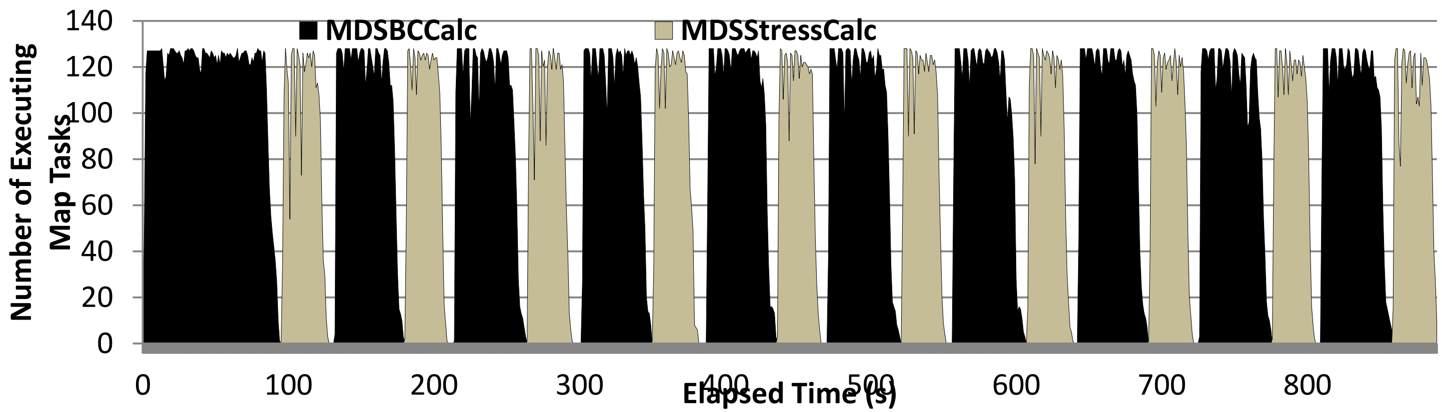
Visualization

- Used PlotViz3 to visualize the 3D plot generated in this project
- It can show the sequence name, highlight interesting points, even remotely connect to HPC cluster and do dimension reduction and streaming back result.

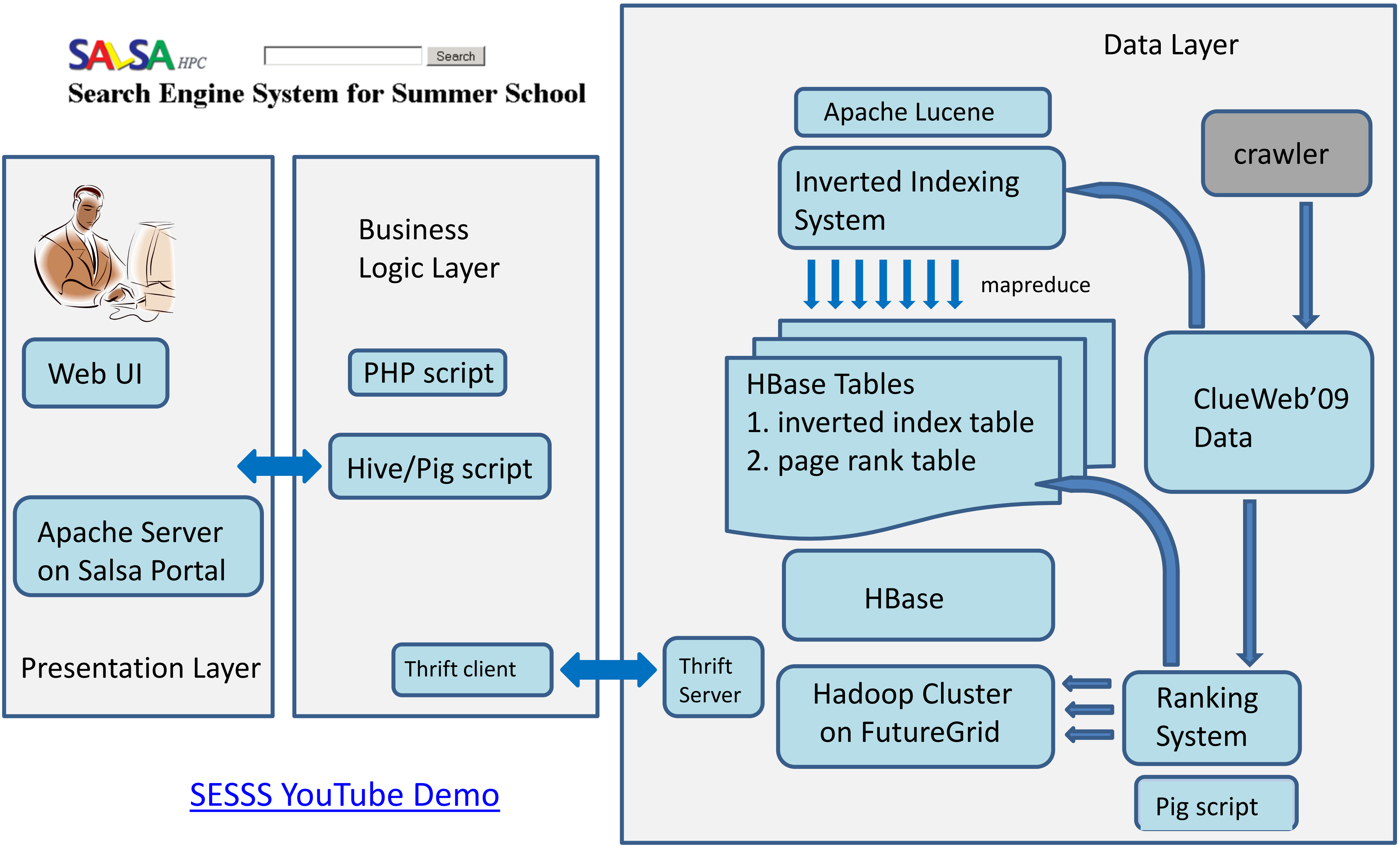


Multi Dimensional Scaling on Azure





Architecture for Search Engine



SESSS YouTube Demo

Parallel Inverted Index using HBase

1. Get inverted index involved in HBase

    "cloud" -> doc1, doc2, …

    "computing" -> doc1, doc3, …

1. Store inverted indices in HBase tables – scalability and availability

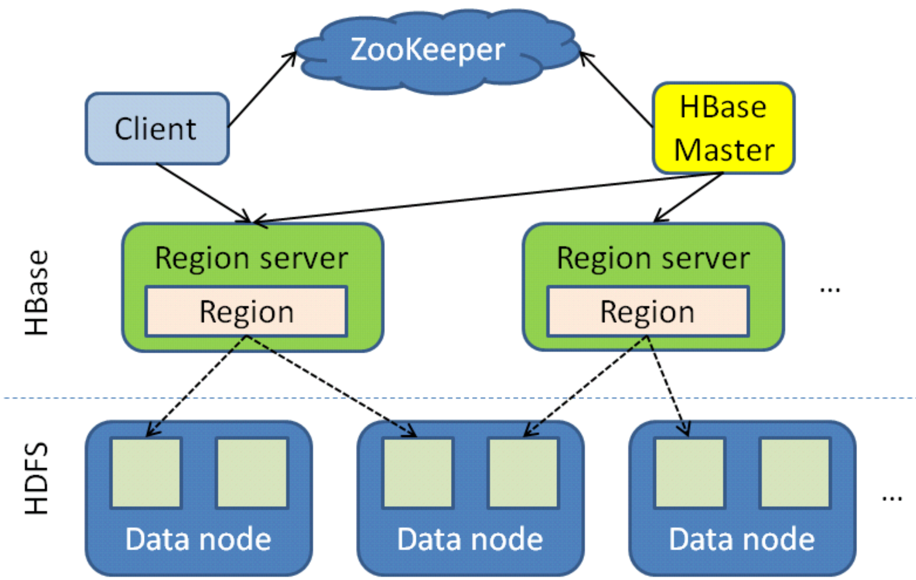
2. Parallel index building with MapReduce (supporting Twister doing data mining on top of this)

3. Real-time document insertion and indexing

4. Parallel data analysis over text as well as index data

5. ClueWeb09 data set for experiments in an HPC environment

HBase architecture:



- Tables split into regions and served by region servers
- Reliable data storage and efficient access to TBs or PBs of data, successful application in Facebook and Twitter
- Problem: no inherent mechanism for field value searching, especially for full-text values

ClueWeb09 dataset

- Whole dataset: about 1 billion web pages in ten languages collected in 2009
- Category B subset:

# of web pages	Language	# of unique URLs	Compressed size	Uncompressed size
50 million	English	4,780,950,903	250GB	1.5TB

- Data stored in .warc.gz files, file size : 30MB – 200MB
- Major fields in a WARC record:

  - HTML header record type, e.g., “response”

  - TREC ID: a unique ID in the whole dataset, e.g., "clueweb09-en0040-54-00000“

  - Target URL: URL of the web page

  - Content: HTML page content

Table schemas in HBase

- Data table schema for storing the ClueWeb09 data set:

	details	
	URI	content
“20000041”	http://some.page.com/index.html	<html> …</html>

- Index table schema for storing term frequencies:

	frequencies		
	“283”	“1349”	… (other document ids)
“database”	3	4	…

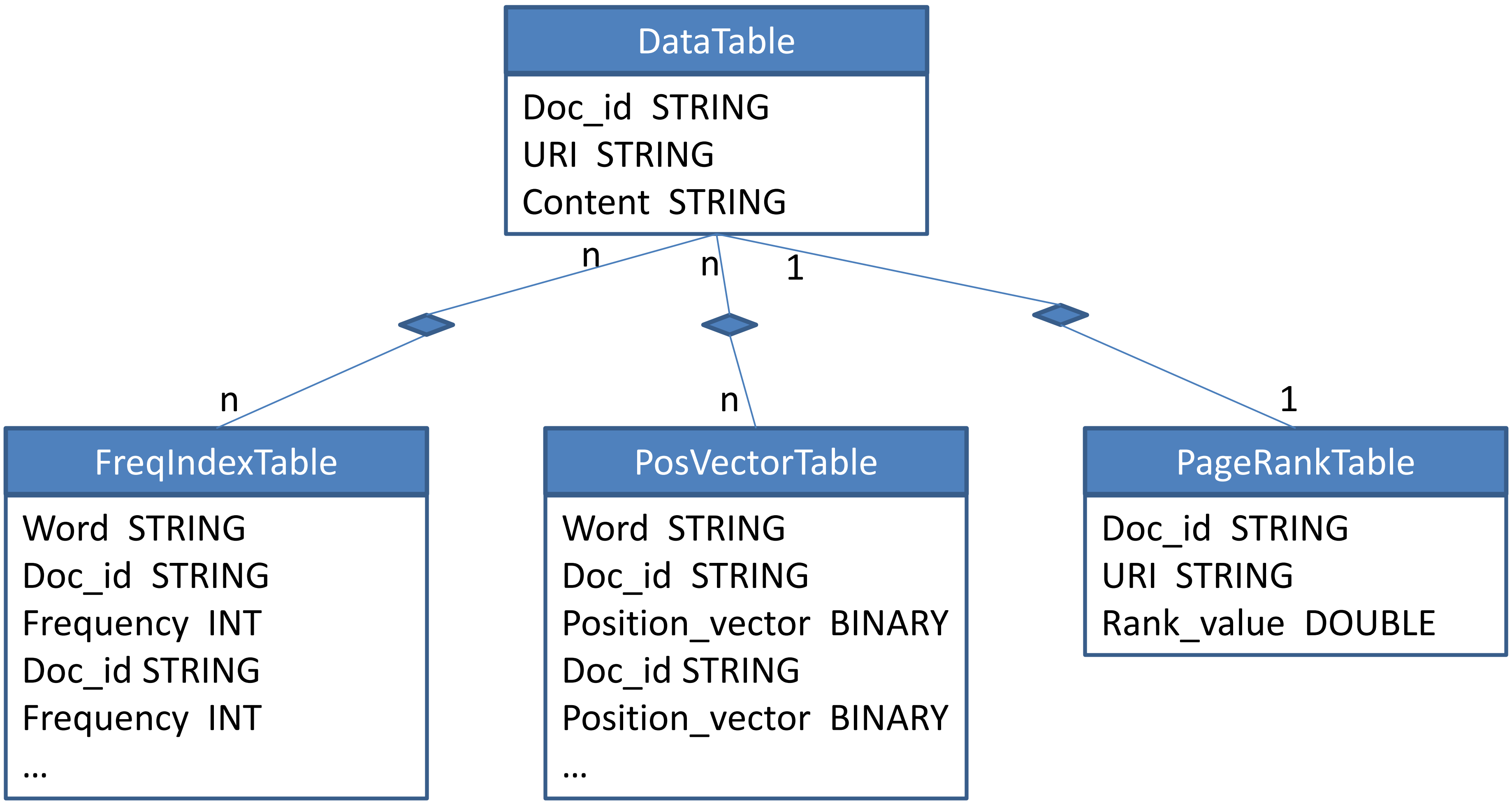
- Index table schema for storing term position vectors:

	positions		
	“283”	“1349”	… (other document ids)
“database”	1, 24, 33	1, 34, 77, 221	…

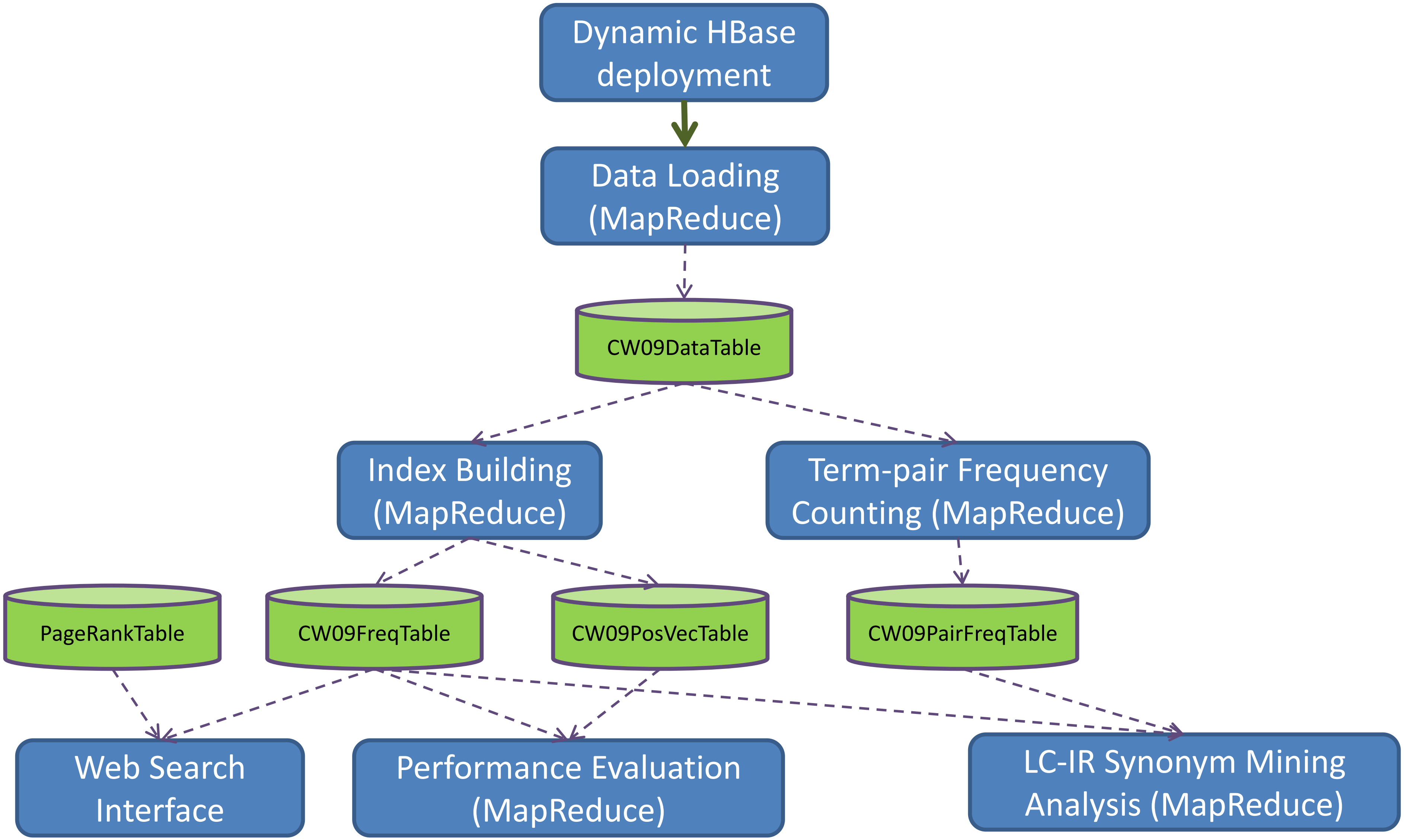
- Table schema for PageRank values:

	PageRanks	
	URI	RankValue
“20000001”	http://en.wikipedia.org/wiki/	43.6

Table schemas – Entity Relation Diagram



System Architecture



LC-IR Synonym Mining

- Mining synonyms from large document sets based on words' co-appearances

$$Similarity_{\mathrm{LC-IR}}(w_1, w_2) = \frac{\min(\mathrm{Hits}(``w_1\ w_2"), \mathrm{Hits}(``w_2\ w_1"))}{\mathrm{Hits}(w_1) \times \mathrm{Hits}(w_2)}$$

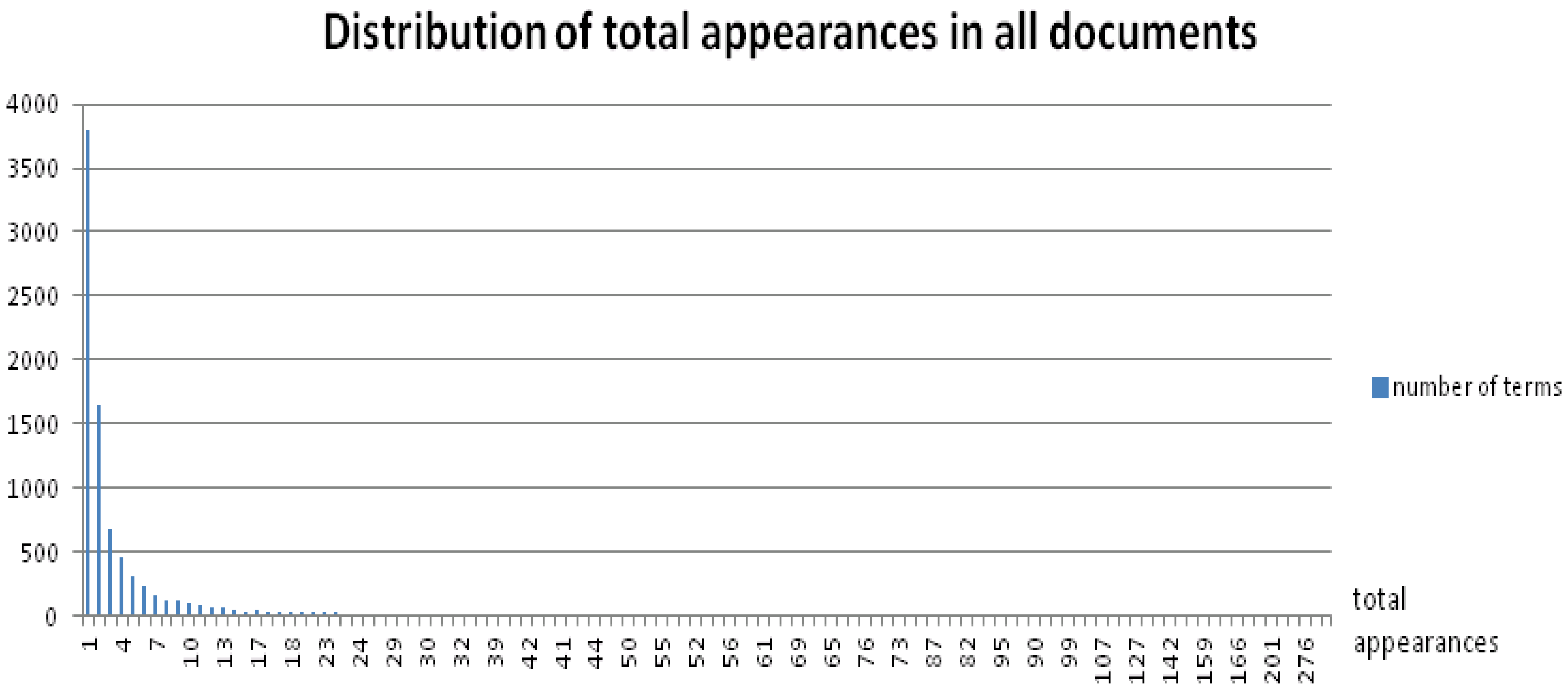
- Steps for completing LC-IR synonym mining in HBase:

  1. Scan the data table and generate a "pair count" table for word-pairs;

  2. Scan the "pair count" table and calculate similarities, looking up single word hits in the index table;

  3. Filter the pairs with similarities lower than a threshold.

Sample Results



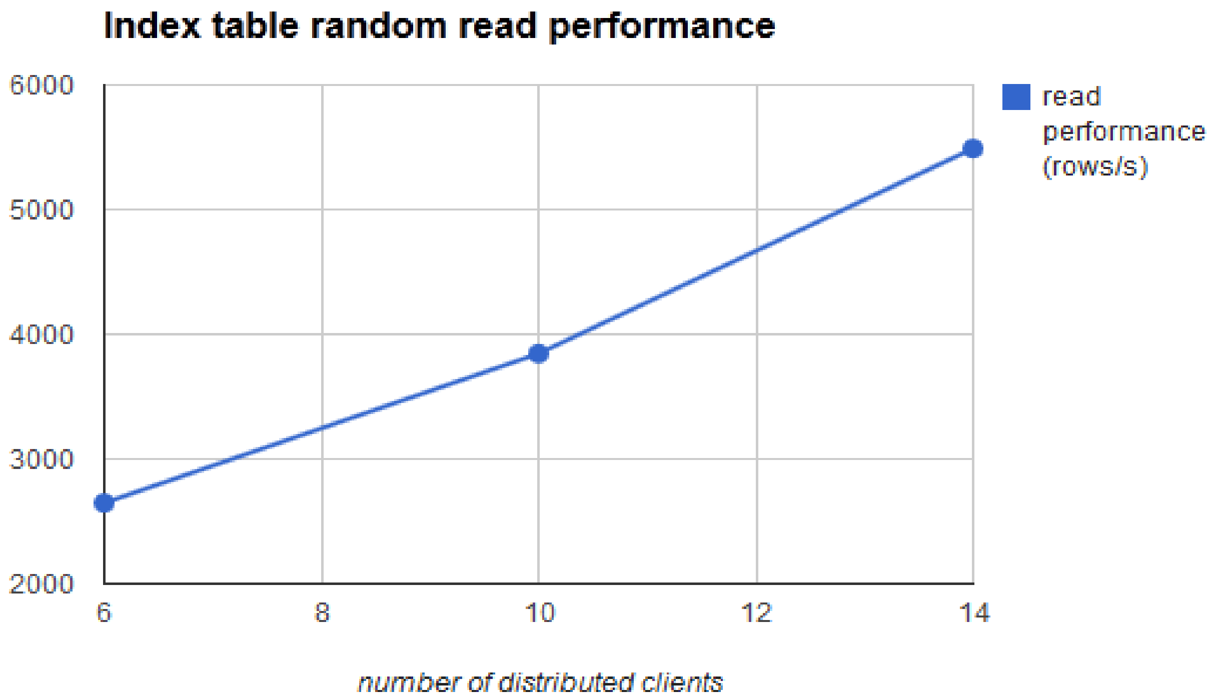
- 100 documents indexed, 8499 unique terms
- 3793 (45%) terms appear only once in all documents
- Most frequent word: “you”

Sample Results

- Example synonyms mined (among 16516 documents):
  - chiropodists podiatrists (0.125, doctors for foot disease)
  - desflurane isoflurane (0.111, narcotic)
  - dynein kinesin (0.111, same type of protein)
  - menba monpa (0.125, a nation/race of Chinese people living in Tibet)
  - lyrica pregabalin (0.125, different names for the same medicine for diabetes)

- Preliminary performance evaluation
  - 6 distributed clients started, each reading 60000 random rows
  - average speed: 2647 rows/s

Sample Results



- Original data table size: 29GB (2,594,536 documents)
- Index table size: 8,557,702 rows (one row for each indexed term)
- Largest row: 2,580,938 cell values, 162MB uncompressed size
- At most 1000 cell values are read from each row in this test
- Aggregate read performance increases as number of concurrent clients increases

Sample Results

Number of nodes	Number of mappers	Index building time (seconds)
8	32	18590
12	37 (15.6% increase)	16142 (15.2% improvement)
16	47 (46.9% increase)	13480 (37.9% improvement)

Index building performance vs. resources increase

- Original data table size: 29GB (2,594,536 documents)
- 6 computing slots on each node
- HBase overhead: data transmission to region servers, cell value sorting based on keys, gzip compression/decompression
- Number of mappers not doubled when number of nodes doubled – because of small table size
- Increase in index building performance is close to increase in number of mappers

Practical Problems and experiences

- Hadoop and HBase configuration

  - Lack of “append” support in some versions of Hadoop: missing data, various errors in HBase and HDFS.

  - Low data locality in HBase MapReduce: “c046.cm.cluster” for Task Tracker vs. “c046.cm.cluster.” for Region Server.

  - Clock not synchronized error: clock not synched with NTP on some nodes.

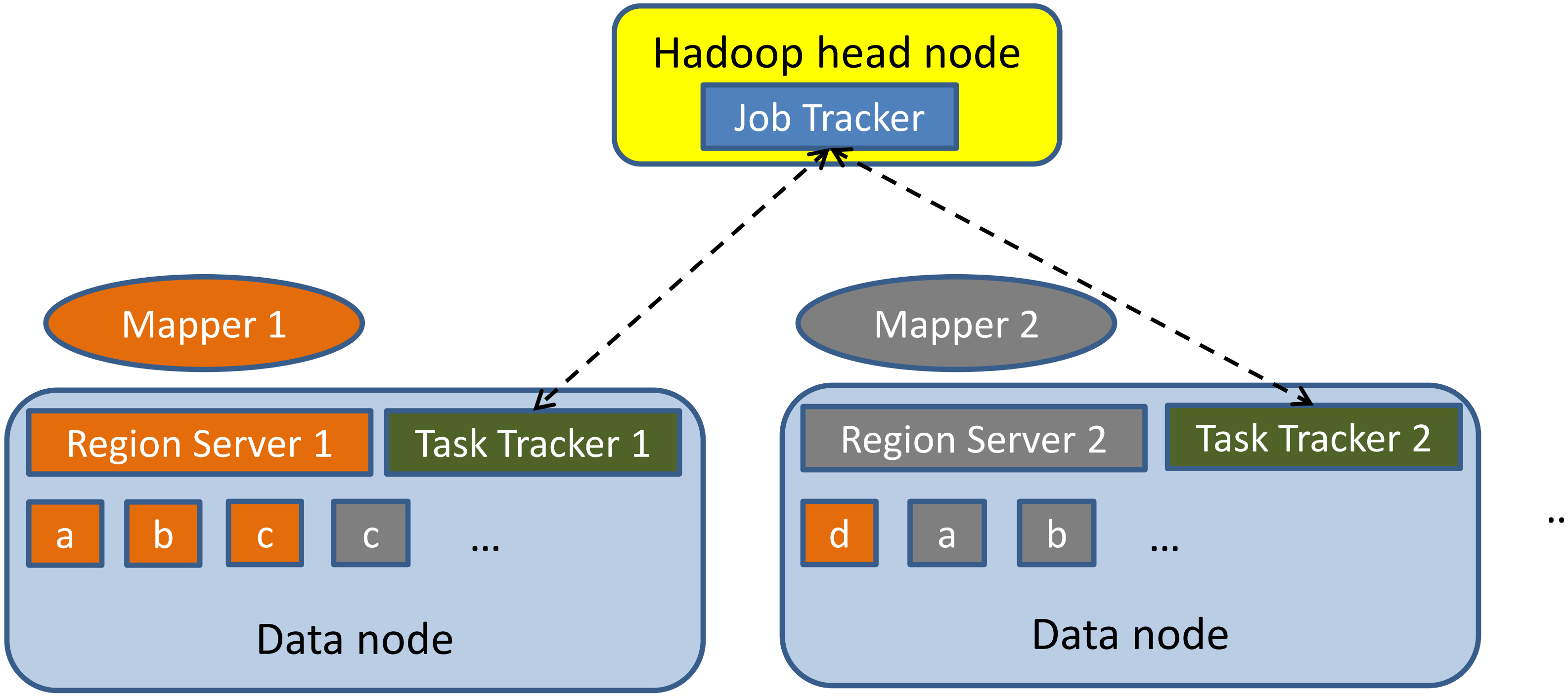
- Optimizations in the synonym mining programs

  - Addition of a word count table with bloom filter.

  - Local combiners for word pair counter.

  - Caching of word counts during the synonym scoring phase.

Low data locality in MapReduce over HBase



- Data splits assigned to mappers by regions (one mapper per region in most cases)
- Mapper deployment based on mapper-region server locality
- Problem: region data blocks not necessarily local to region servers
- Data locality gets even worse after region splits or region server failures

Ackowledgements

SALSA HPC Group

http://salsahpc.indiana.edu

School of Informatics and Computing

Indiana University

Microsoft

INDIANA UNIVERSITY

NSF

NATIONAL INSTITUTES OF HEALTH

Future Grid