Distributed Dual Iterative Pattern Relation Expansion (D-DIPRE)

Anup Chalamalla Mina Farid

Outline

- DIPRE
  - Introduction
  - Problem Definition
  - Algorithm
- Distributed DIPRE
  - Basic idea
  - Implementing different modules
  - Data flow
  - Iterative MapReduce
- Conclusions and Future Work

DIPRE- Introduction and Problem Definition

- Large amount of information on the Web
- Extract structured information from *unstructured documents*

- *D* large set of documents (WWW)
- Looking for *occurrences* of *R*
- *R* is a binary relation, e.g., (author, title) , (person_name, email)

DIPRE- Algorithm

1. Feed some seeds
2. Find Occurrences
3. Generate Patterns
4. Find patterns matches
5. If enough tuples are found exit, else repeat 2

DIPRE- Algorithm (cont'd)

1. **Feed some seeds**
2. Find Occurrences
3. Generate Patterns
4. Find patterns matches
5. If enough tuples are found exit, else repeat 2

DIPRE- Algorithm (cont'd)

1. **Feed some seeds**

Provide some sample instances of the relation

For example,

Author	Book Title
Isaac Asimov	The Robots of Dawn
David Brin	Startide Rising
James Gleick	Chaos: Making a New Science
Charles Dickens	Great Expectations
William Shakespeare	The Comedy of Errors

DIPRE- Algorithm (cont'd)

1. Feed some seeds
2. **Find Occurrences**
3. Generate Patterns
4. Find patterns matches
5. If enough tuples are found exit, else repeat 2

DIPRE- Algorithm (cont'd)

2. Find Occurrences of Seeds

- Occurrence Structure: (prefix, author, middle, book, suffix, order, url)
- Example:
  - look for (Charles Dickens, Great Expectations) in the domain www.books.com

    www.books.com/TopRated

    "The famous writer Charles Dickens wrote Great Expectations book"
  - Extracted Occurrence:

    (The famous writer, Charles Dickens, wrote, Great Expectations, book, true, www.books.com/TopRated)

- Repeat for all seeds in all documents
- Result: A set of occurrences of seeds in the documents

DIPRE- Algorithm (cont'd)

1. Feed some seeds
2. Find Occurrences
3. **Generate Patterns**
4. Find patterns matches
5. If enough tuples are found exit, else repeat 2

DIPRE- Algorithm (cont'd)

3. Generate Patterns

Pattern Structure: (order, urlprefix, prefix, middle, suffix)

- Group occurrences having similar "order" and "middle"

(The famous writer, Charles Dickens, **wrote**, Great Expectations, book, **true**, www.books.com/TopRated)

(The great writer, Nicholas Sparks, **wrote**, The Last Song , book, **true**, www.books.com/BestSellers)

Generate a pattern as general as possible to match all occurrences.

writer **.*?** wrote **.*?** book

Prefix = writer Suffix = book order = true

Middle = wrote urlprefix = www.books.com

DIPRE- Algorithm (cont'd)

1. Feed some seeds
2. Find Occurrences
3. Generate Patterns
4. **Find patterns matches**
5. If enough tuples are found exit, else repeat 2

DIPRE- Algorithm (cont'd)

4. Find patterns matches and extract relations

writer **.*?** wrote **.*?** book

.....The writer Mario Puzo wrote The Godfather book.....

Extract relation (Mario Puzo, The Godfather)

DIPRE- Algorithm (cont'd)

5. If enough tuples are found exit, else repeat

repeat again, having new tuples as seeds

New Seed: (Mario Puzo, The Godfather)

.... the book The Godfather was written by Mario Puzo....

	Prefix	Author	Middle	Book	Suffix	Order	URL
Occurrence:	(the book,	Mario Puzo,	was written by,	The Godfather,	NULL,	false,	www.library.com)

	order	urlprefix	prefix	middle	suffix
Pattern:	(False,	library.com,	The book,	was written by,	NULL)

the book **.*?** was written by **.*?**

Match patterns ....

DIPRE- End Result

Output:

1. Tuples of Relation R extracted (set of authors and book titles)
2. List of Patterns to extract books

- Patterns are used to extract relations from new documents added to the database

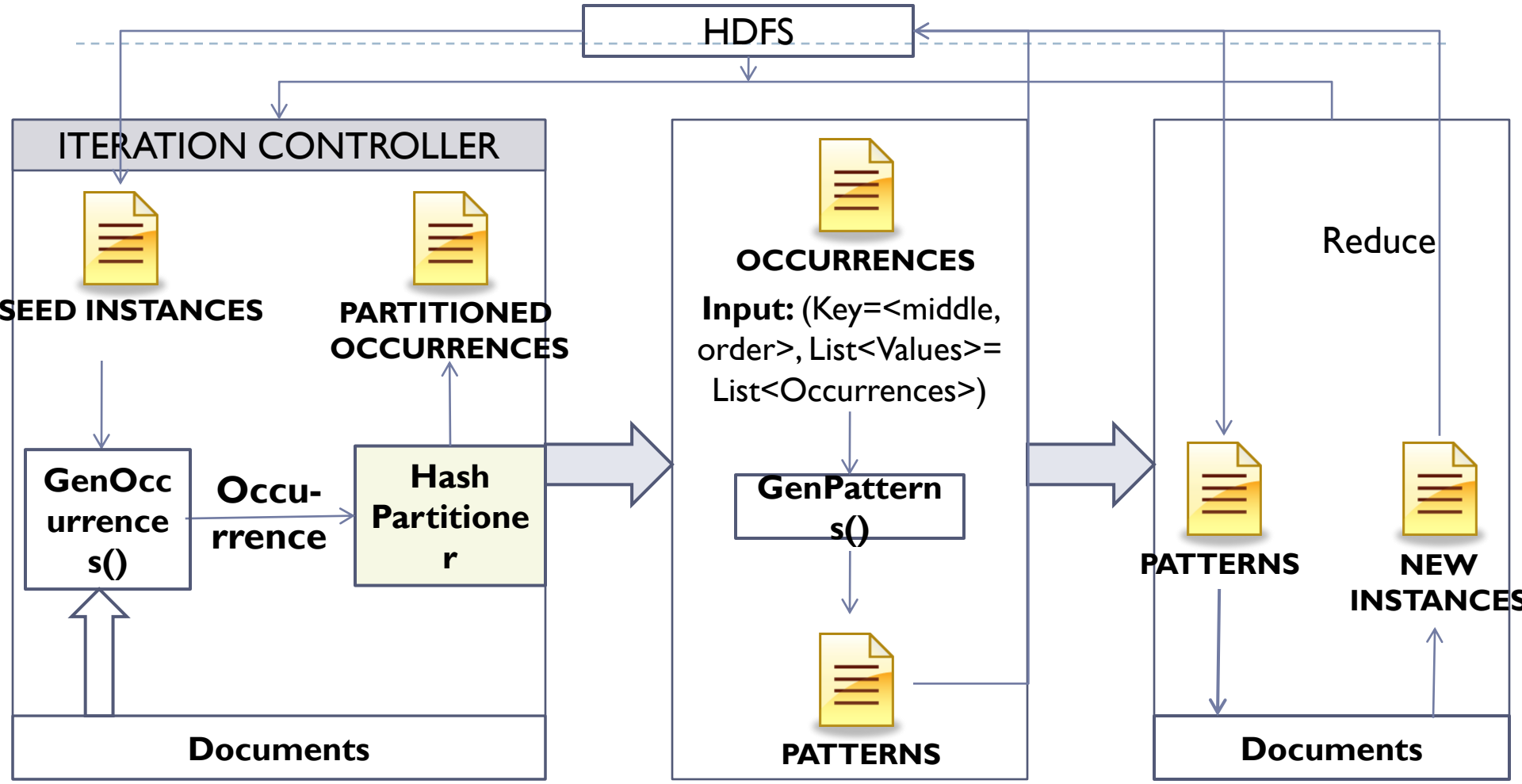
Distributed DIPRE System

- → {Instances of a Relation} $\rightarrow_D$ {Occurrences in the documents} → {RegEx. Patterns} $\rightarrow_D$ {New instances}
- All documents and seed instances reside on Hadoop's HDFS
- Hadoop's MapReduce framework is used to process instances and patterns over documents local to each map and reduce worker machines on HDFS
- New instances and patterns generated in each iteration are stored on HDFS

Modules in the implementation

- Document Loading – Hadoop loads the documents' corpus and places the documents on HDFS on the name node and data nodes in the cluster.
- Occurrence Extractor – Finding occurrences of seed instances in the documents and extracting their context {prefix, suffix, middle, order}
  - Map function:: [(key=docID, value=documentText) $\rightarrow_R$ (key={middle, order}, value=Occurrence)]
- Pattern Extractor – Extracts patterns on to HDFS from re-grouped occurrences
  - Reduce function:: [(key={middle, order}, value=List<Occurrence>) → List<Patterns>]
- Pattern Matcher — Match extracted patterns against documents and output more instances which are again fed as seeds to the occurrence extractor
  - Map function:: [(key=docID, value=documentText) $\rightarrow_P$ List<R>]
- Iteration Controller – Stops the iterative process of pattern-relation extraction when it finds that number of new seed instances generated in an iteration is very small.

Data flow



Occurrence Extractor (Map function)

Pattern Extractor (Reduce Function)

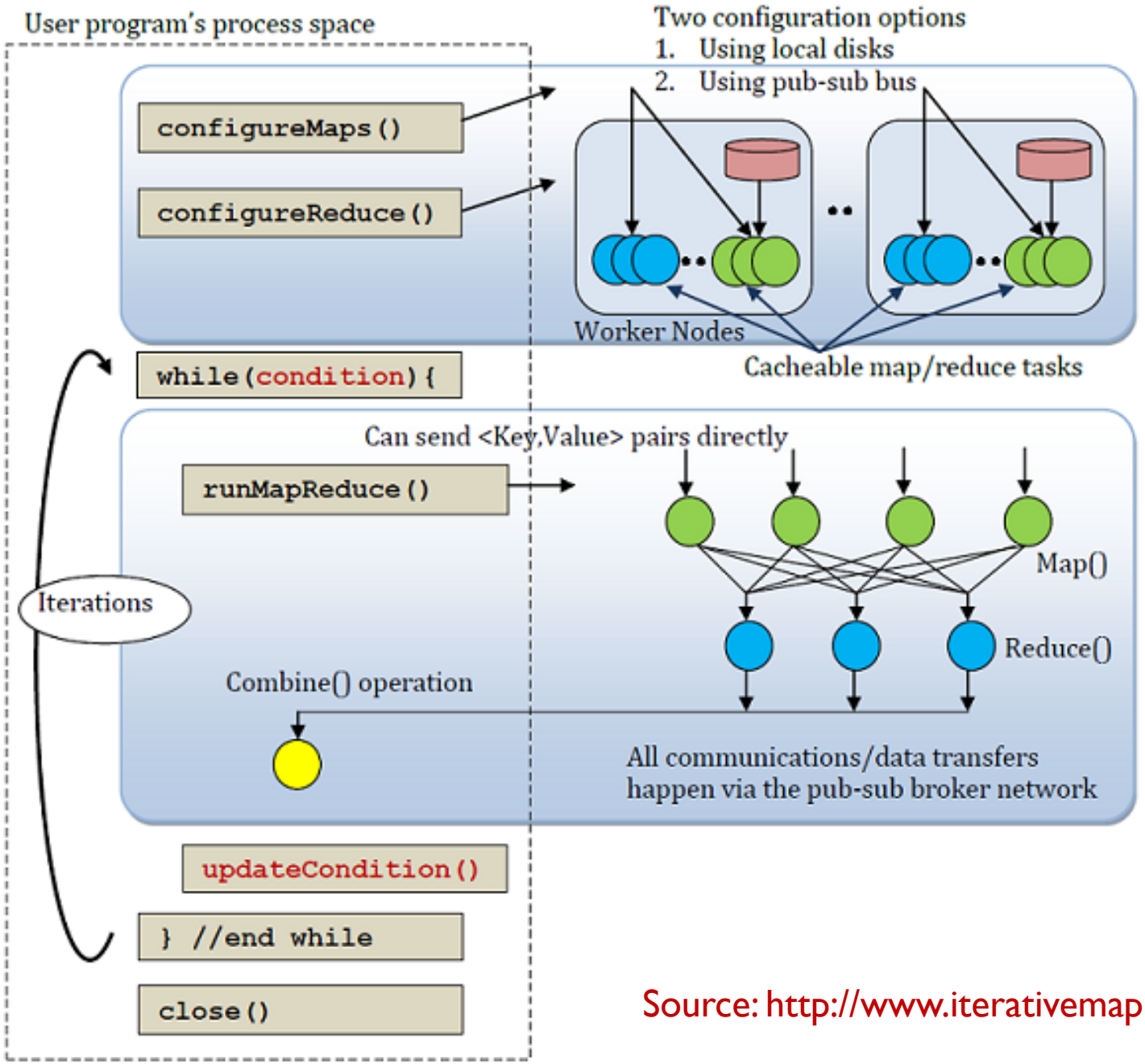
Pattern Matcher (Map Function)

Status of Implementation

- Implemented the DIPRE system with map and reduce functions for various stages in the processing
- Need to configure Hadoop to run the map reduce jobs iteratively
- Need to configure Hadoop to run on Amazon EC2 and produce scalability results

Iterative Map Reduce

- Additional Combine phase to collect all the final output instances of an iteration
- Distinction on static and variable data
- Remove or rename the instances file and load new instances for next iteration
- Hadoop supports iteratively running Map Reduce tasks



Source: http://www.iterativemapreduce.org/

Conclusion

- A distributed information extraction system, which is logically same as DIPRE
  - Can extract instances of an arbitrary relation from documents of any arbitrary domain
- Experiments: Scalability results on
  - Number of nodes used (vs) Time taken
  - Number of documents processed (vs) Time taken,
- Systems Compared: DIPRE, D-DIPRE

Future Work

- Use inverted index to process documents for generating occurrences and patterns
  - Katta: Distributed Lucene system which can create a distributed index on Hadoop

References

- Brin, Sergey (1999) *Extracting Patterns and Relations from the World Wide Web*. Technical Report. Stanford InfoLab. (Publication Note: WebDB Workshop at EDBT'98)

- S. Khaitan, G. Ramakrishnan, S. Joshi, Anup K. Chalamalla: *RAD: A Scalable Framework for Annotator Development*. ICDE 2008.