MAPREDUCE

FUNCTIONAL PROGRAMMING

- Map
  - Apply function to transform elements of a list
  - Return results as a list
- Reduce
  - Apply function to all elements of a list
  - Collect results and return as a single value

DEFINITION

- MapReduce: A software framework to support processing of massive data sets across distributed computers

SAMPLE USE CASE

- Back end credit card processor
- Nightly processing of millions of transactions
- Processing requires grouping, sorting, and merchant wide analysis
  - Can't just divide the over all list into equal parts as further analysis is necessary
- Tight processing window

DESCRIPTION

- Simple, powerful programming model
- Language independent
- Can run on a single machine, but shines for distributed computing and extreme datasets
- Break down the processing problem into embarrassingly parallel atomic operations

ALGORITHM

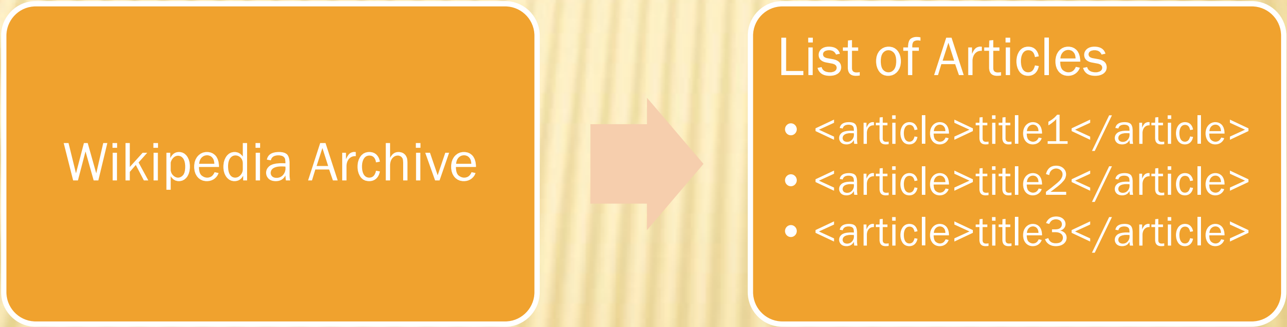
- Map Phase
  - Raw data analyzed and converted to name/value pair
- Shuffle Phase
  - All name/value pairs are sorted and grouped by their keys
- Reduce Phase
  - All values associated with a key are processed for results

MAPREDUCE WALK THROUGH

- Goal: Construct a word frequency of all the words in Wikipedia

STEP 0: SPLIT DATA

- Raw input data divided into N parts
  - N > number of machines
- Split must be context specific



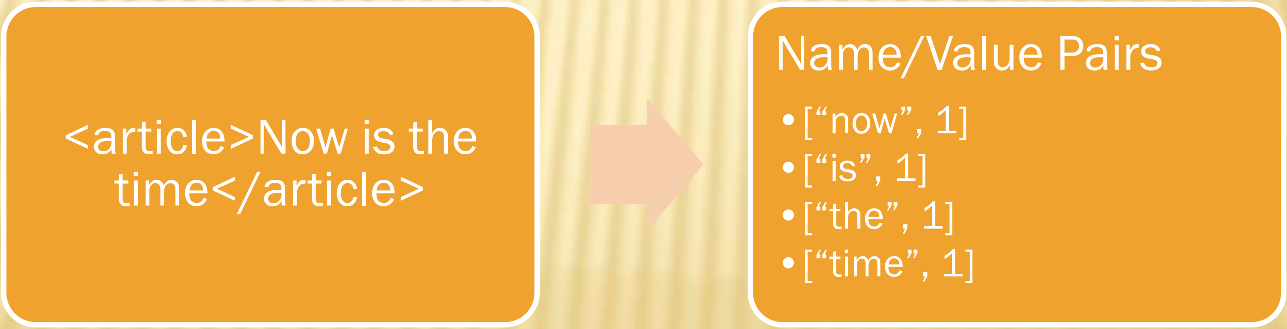
Wikipedia Archive

List of Articles

- <article>title1</article>
- <article>title2</article>
- <article>title3</article>

STEP 1: MAP

- Each machine takes/receives a single slice of the raw input for mapping
- The map function processes the input file and emits a name/value pair of the relevant data



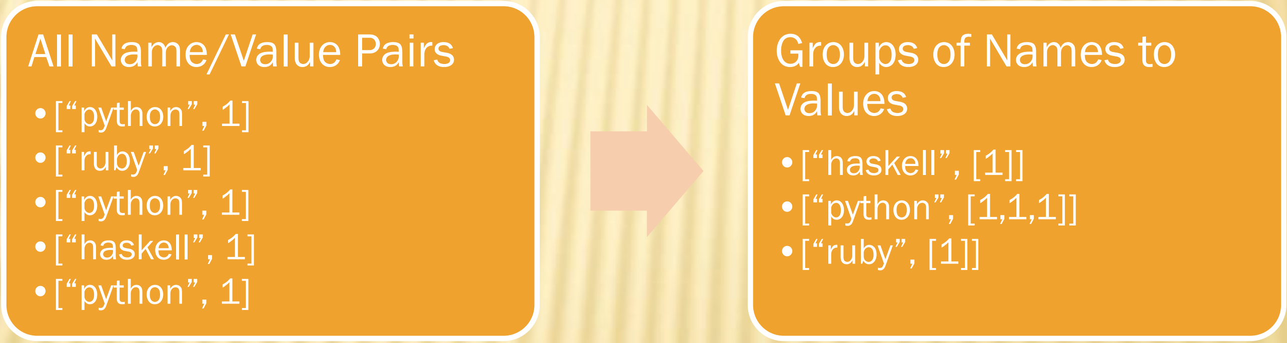
<article>Now is the time</article>

Name/Value Pairs

- [“now”, 1]
- [“is”, 1]
- [“the”, 1]
- [“time”, 1]

STEP 2: SHUFFLE

- The results of the map phase are sorted and grouped by the key in each key value pair.



All Name/Value Pairs

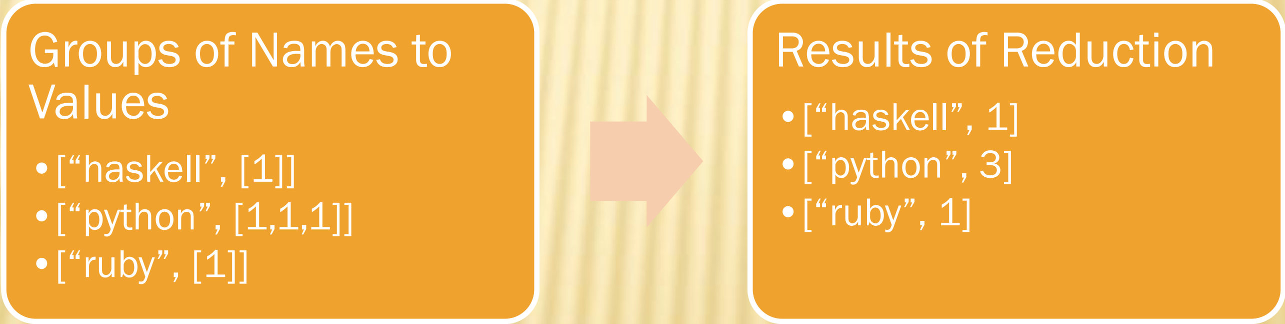
- [“python”, 1]
- [“ruby”, 1]
- [“python”, 1]
- [“haskell”, 1]
- [“python”, 1]

Groups of Names to Values

- [“haskell”, [1]]
- [“python”, [1,1,1]]
- [“ruby”, [1]]

STEP 3: REDUCE

- Results from shuffle phase divided into M parts
  - M > number of machines
- Each machine runs a reduction method on a part of shuffle results.



Groups of Names to Values

- [“haskell”, [1]]
- [“python”, [1,1,1]]
- [“ruby”, [1]]

Results of Reduction

- [“haskell”, 1]
- [“python”, 3]
- [“ruby”, 1]

MAPREDUCE BENEFITS

- Scale
  - Processing speed increases with number of machines involved
- Reliable
  - Loss of any one machine doesn’t stop processing
- Cost
  - Often built from heterogeneous commodity grade computers

USE CASE RESULTS

- Processing time of 1 million records
  - Originally ~3 hours
  - Reduced to 40 minutes on 5 computers

OTHER MAPREDUCE INSTALLATIONS

- Google – Index building
- Visa – Transaction Processing
- Facebook – Facebook Lexicon
- Intelligence Community
- Yahoo/Google – Terabyte Sort
  - 10 billion, 100 byte records
  - Yahoo: 910 nodes, 206 seconds
  - Google: ~1,000 nodes, 68 seconds

QUESTIONS