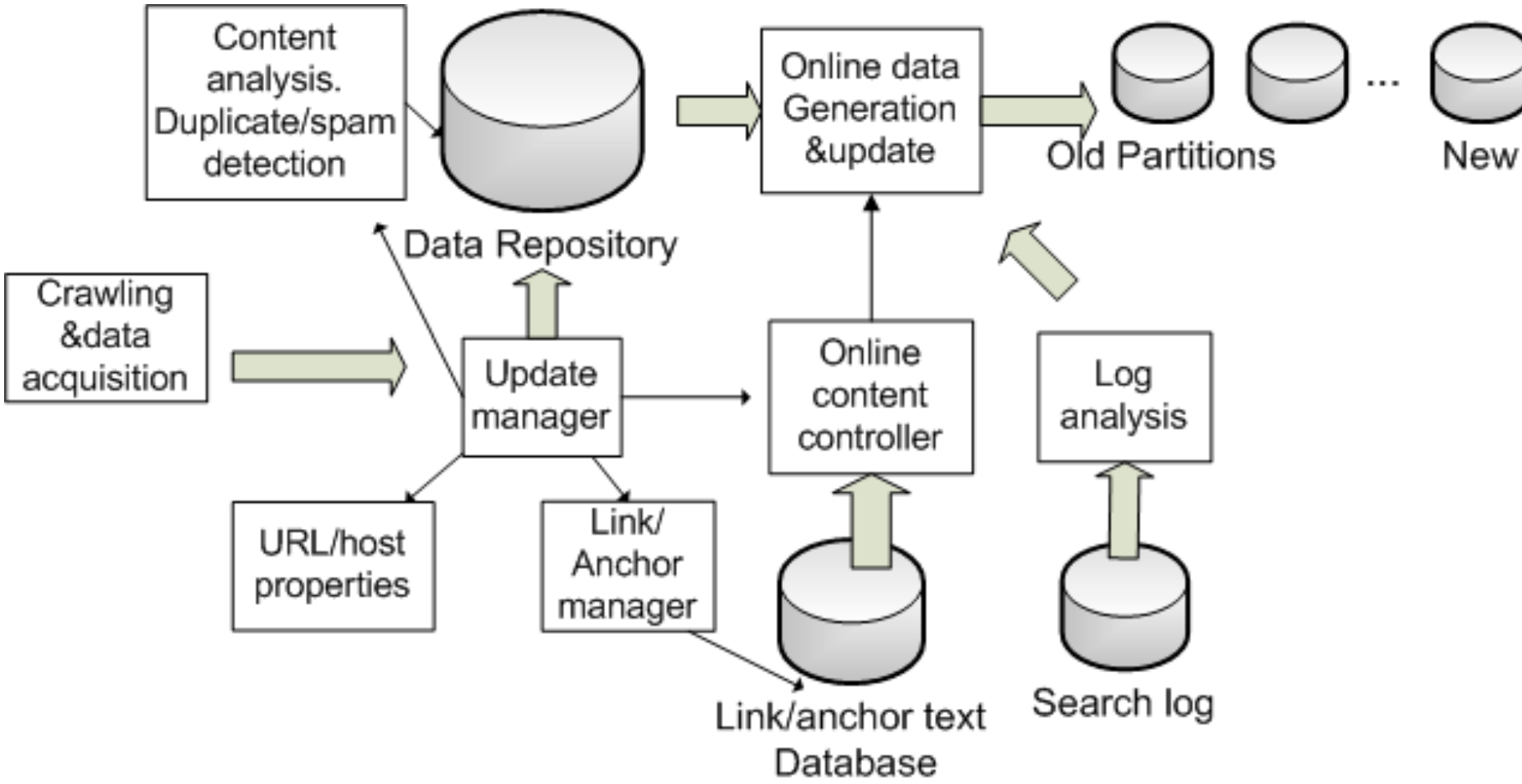
Offline Data Processing: Tasks and Infrastructure Support

T. Yang, UCSB 293S, 2020

Table of Content

- **Offline incremental data processing: case study**
  - Content management for large index
  - Text mining, knowledge graph
  - Example of content analysis
- **Duplicate content removal**
- **System support for offline data processing**

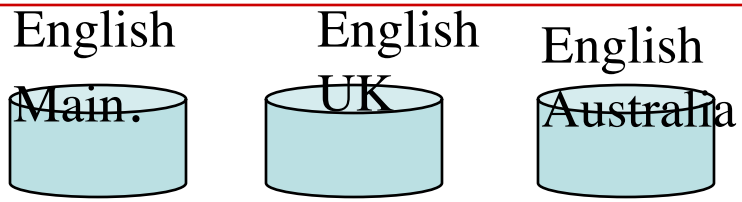
Offline Architecture for Ask.com Search



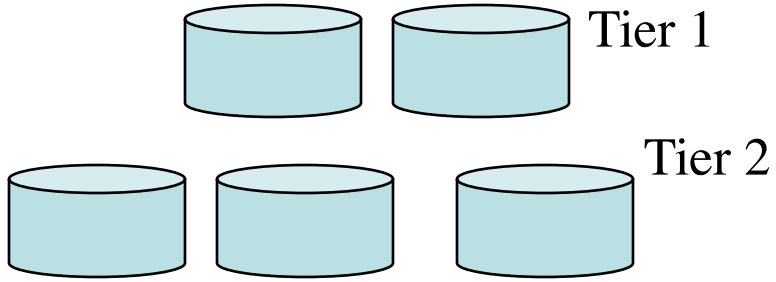
Content Management

- **Organize the vast amount of pages crawled to facilitate online search.**
  - Data preprocessing
  - Inverted index
  - Compression
  - Classify and partition data
- **Collect additional content and ranking signals.**
  - Link, anchor text, log data
- **Extract and structure content**
- **Duplicate detection**
- **Anti-spamming**

Classifying and Partitioning data



- **Classify**
  - Content quality. Language/country etc
- **Partition**
  - Based on languages and countries. Geographical distribution based on data center locations
  - Partition based on quality
    - First tier --- high chance that users will access
      - Quality indicator
      - Click feedback
    - Second tier – lower chance



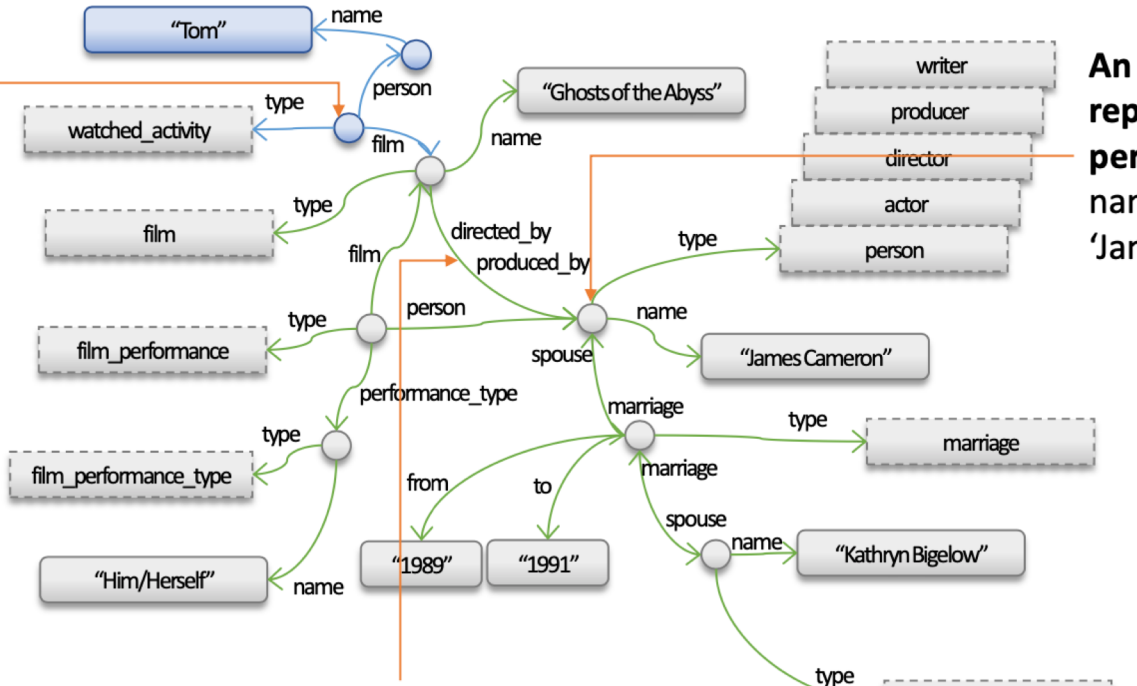
Text mining

- **“Text mining” is a cover-all marketing term**
- **A lot of what we’ve already talked about is actually the bread and butter of text mining:**
  - Text classification, clustering, and retrieval
- **But we will focus in on some of the higher-level text applications:**
  - Extracting document metadata
  - Entities/knowledge graphs
  - Topic tracking and new story detection
  - Cross document entity and event coreference
  - Text summarization

Knowledge Graph

A knowledge graph is represented as entities, edges and attributes

Personal entity showing that Tom watched Ghosts of the Abyss



An entity representing a person with name **attribute** 'James Cameron'

Edge (i.e. relationship) showing that "Ghosts of the Abyss" was 'directed_by' and 'produced_by' James Cameron

Key concepts

Entity	Represent something in the real world
Edge	Represent relationship
Attribute	Represent something about an entity
Ontology	Definition of possible types of entities, relationships and attributes

Knowledge Graphs and Challenges

General knowledge graphs

- Freebase Wikidata, Dbpedia
- Google Knowledge Vault, Google KG, Microsoft Satori KG

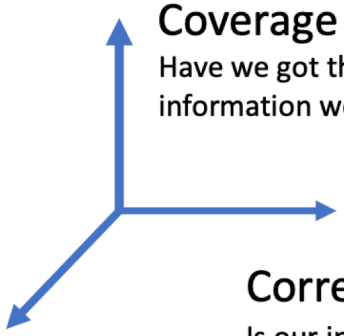
Large vertical KGs

- Facebook (social network), LinkedIn (people graph)
- Amazon (product graph)

Challenges for building/maintaining a scalable large KG

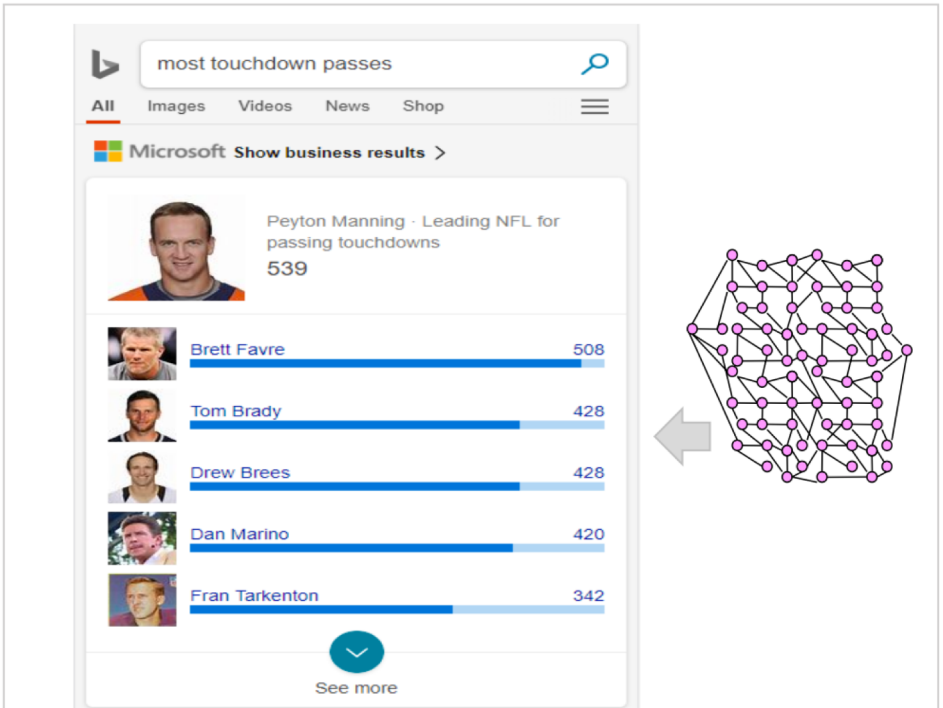
2B+ entities
130B+ Web pages
44+ languages

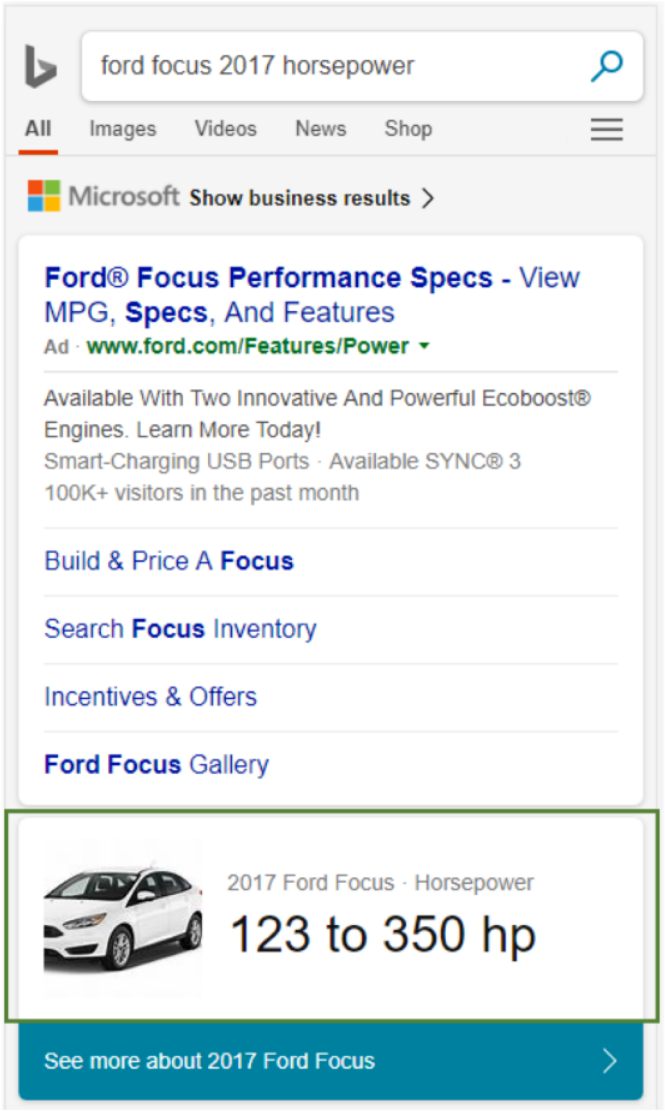
Will Smith: Single entity, 108K facts assembled from 41 web sites.
There are 200 Will Smiths on Wikipedia alone.



Usage of Knowledge Graphs for Search and Other Information Systems

- **Search and NLP questions**
  - Give direct answers
  - Enhance ranking
- **Recommendation**
- **Auto conversation**





Information extraction to enhance information on web pages and refine knowledge graphs

- **Getting semantic information out of textual data**
  - Understand more information on web pages
  - Refine knowledge entities and extract their relationship
  - Validation, association, duplicate removal/merging (entity linking), error correction, content refreshing
- **Look for specific types of web pages:**
  - E.g. an event web page:
    - What is the name of the event?
    - What date/time is it?
    - How much does it cost to attend
  - Home pages for persons, organizations,
- **Many vertical domains: resumes, health, products, …**

Examples of Context Extraction/Analysis

- **Getting semantic information out of textual data**
  - Identify key phrases that capture the meaning of this document. For example, title, section title, highlighted words.
  - Identify parts of a document representing the meaning of this document.
    - Many web pages contain a side-menu, which his less relevant to the main content of the documents
  - Identify entities and their relationships, attributes
  - Capture page content through Javascript analysis.
    - Page rendering and Javascript evaluation within a page

Example of Content Analysis

- **Identify content block related to the main content of a page**
  - Non-content text/link material is de-prioritized during indexing process

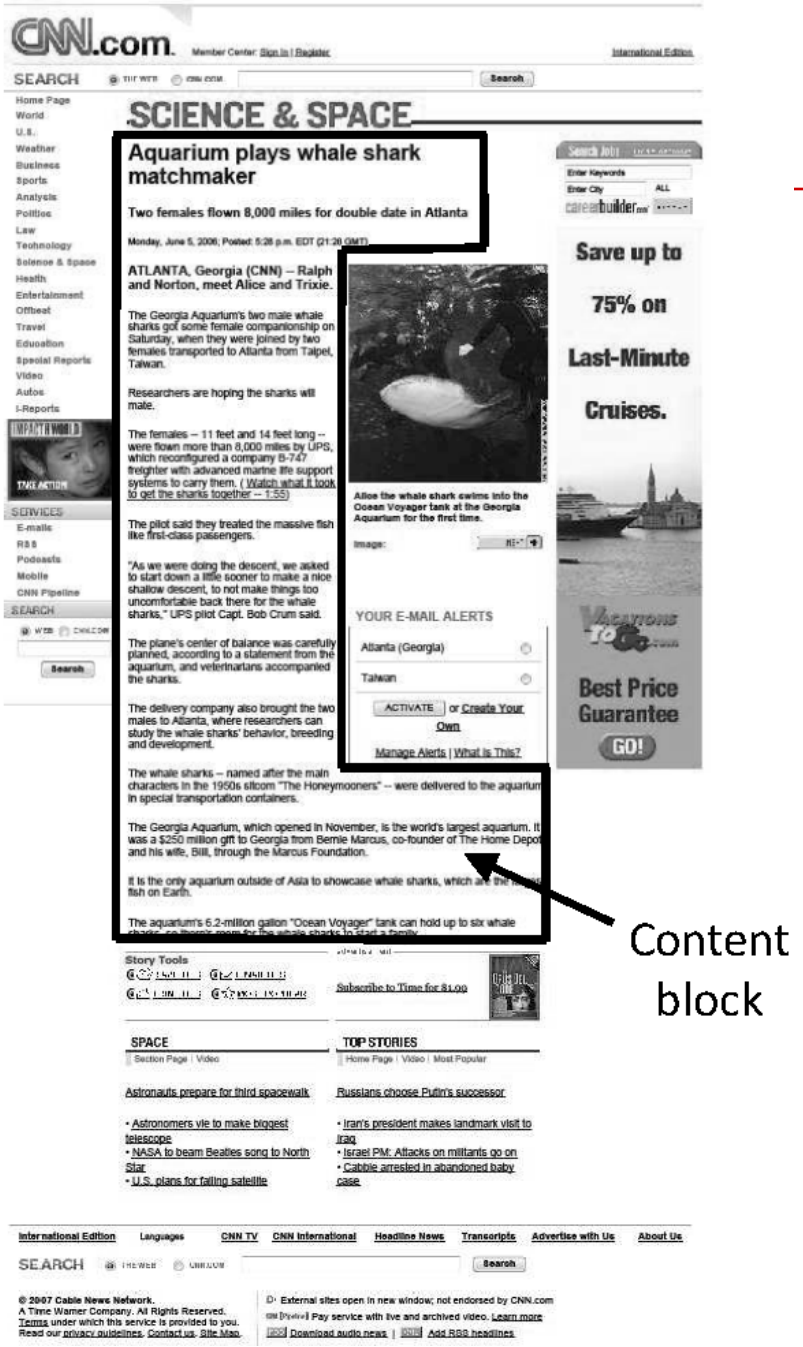
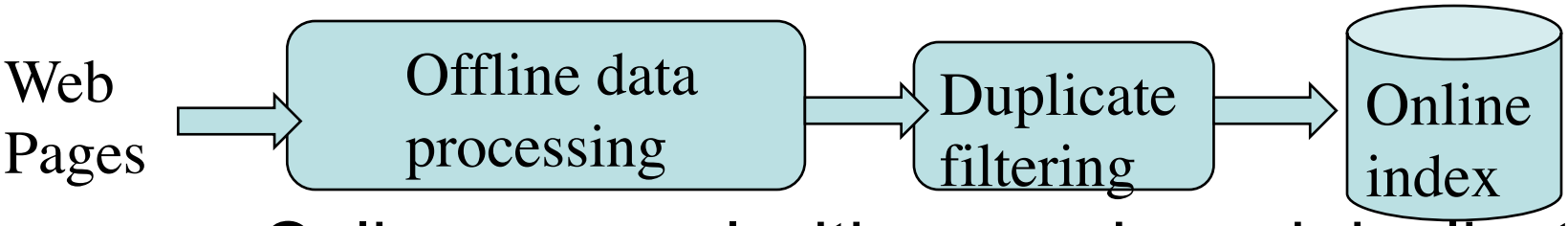


Table of Content

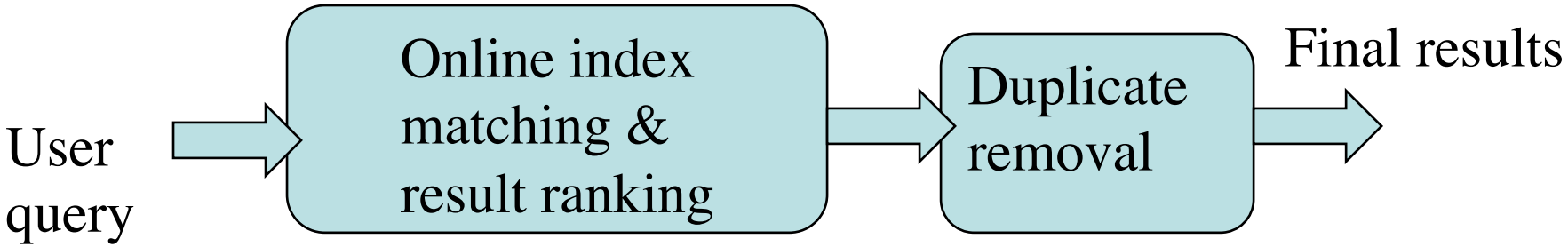
- **Offline incremental data processing: case study**
  - Content management for large index
  - Text mining, knowledge graph
  - Example of content analysis
- **Duplicate content removal** 
- **System support**

Redundant Content Removal in Search Engines

- **Over 1/3 of Web pages crawled are near duplicates**
- **When to remove near duplicates?**
  - Offline removal

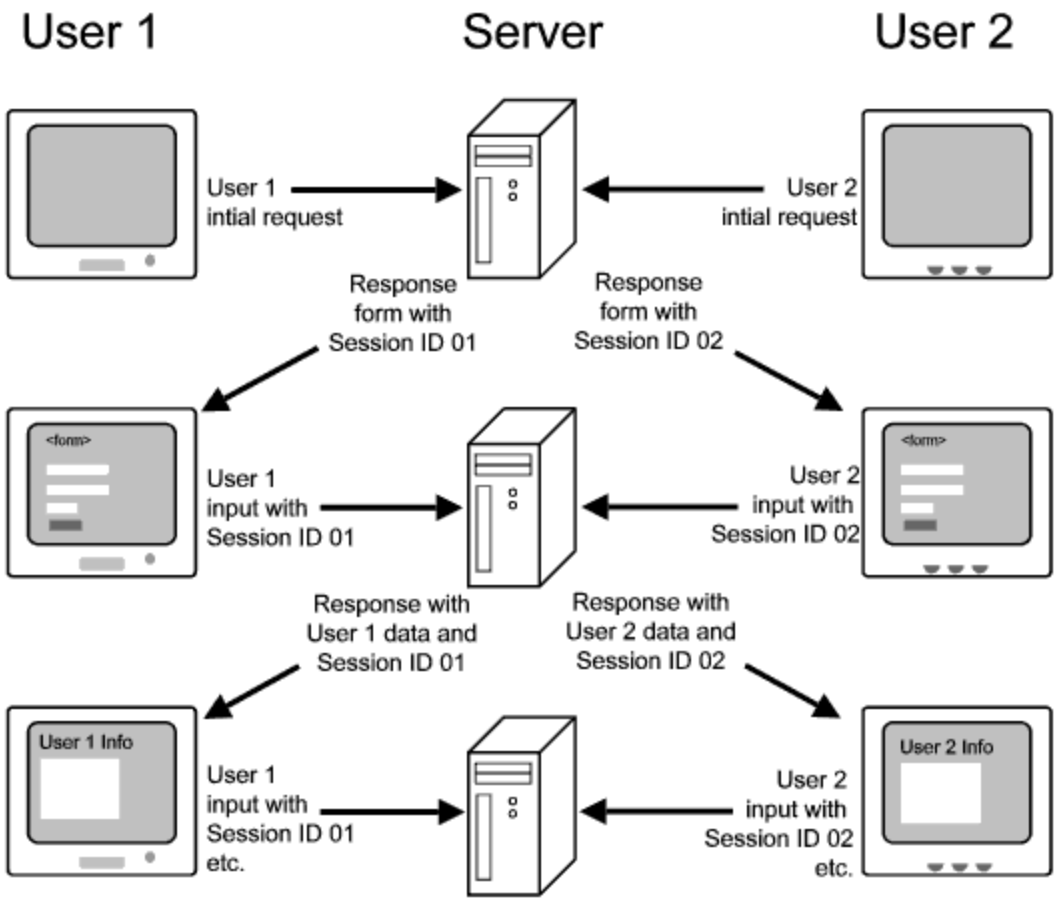


  - Online removal with query-based duplicate removal



Why there are so many duplicates?

- **Same content, different URLs, often with different session IDs.**
- **Crawling time difference**

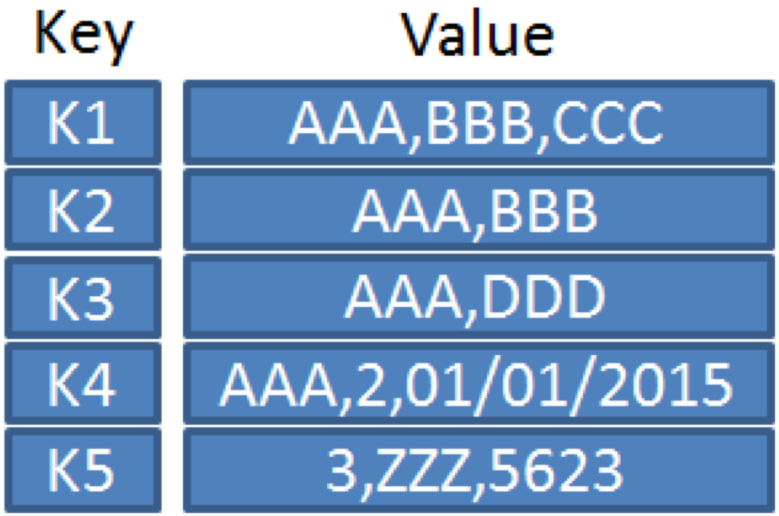


Tradeoff of online vs. offline removal

	Online-dominating approach	Offline-dominating approach
Impact to offline data processing design	High precision Low recall Remove fewer duplicates	High precision High recall Remove most of duplicates Higher offline burden
Impact to online system design	More burden to online deduplication	Less burden to online deduplication
Impact to overall cost	Higher serving cost	Lower serving cost

Key Value Stores/Storage

- **Handle huge volumes of data, e.g., PetaBytes!**
  - Store (key, value) tuples



- **Simple interface**
  - `put(key, value);` Insert/write "value" associated with "key"
  - `value = get(key);` Get/read data associated with "key"

Used sometimes as a simpler but more scalable "database"

Key Values: Examples

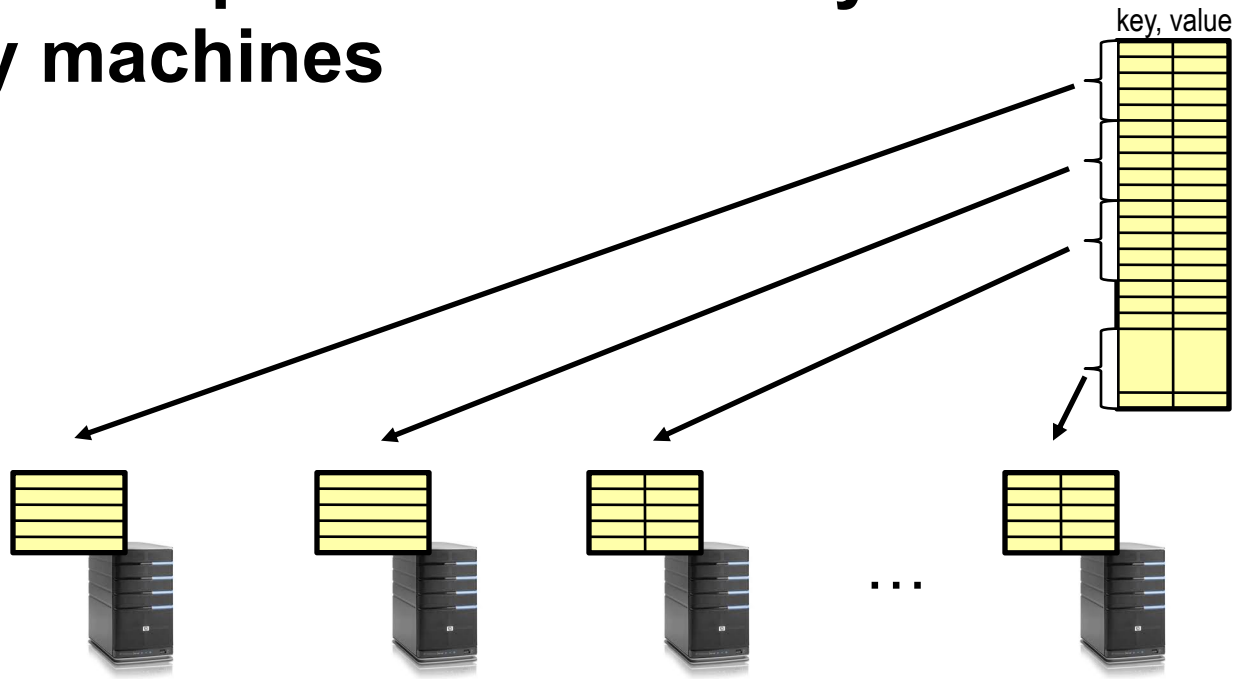
- **Web search: store documents, cache results, store URL properties**
  - Document server, image server, cache server, URL server
- **Amazon shopping:**
  - Key: customerID
  - Value: customer profile (e.g., buying history, credit card, ..)
- **Facebook, Twitter accounts:**
  - Key: UserID
  - Value: user profile (e.g., posting history, photos, friends, …)
- **iCloud/iTunes:**
  - Key: Movie/song name
  - Value: Movie, Song

Key-Value Storage Systems in Real Life

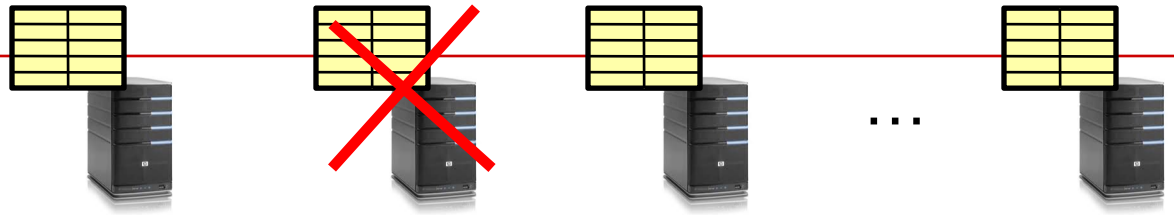
- **Amazon**
  - DynamoDB: internal key value store used for Amazon.com (cart)
  - Amazon SimpleDB. Simple Storage System (S3)
- **BigTable/HBase/Hypertable:** distributed, scalable data store
- **Cassandra:** "distributed data management system" (developed by Facebook)
- **Memcached:** in-memory key-value store for small chunks of arbitrary data (strings, objects)
- **BitTorrent distributed file location:** peer-to-peer sharing system
- **Redis, Oracle NSQL Database…**
- **Distributed file systems: set of (file block ID, file block)**

Key Value Store on a Cluster of Machines

- Also called Distributed Hash Tables (DHT)
- Main idea: partition set of key-values across many machines



Challenges



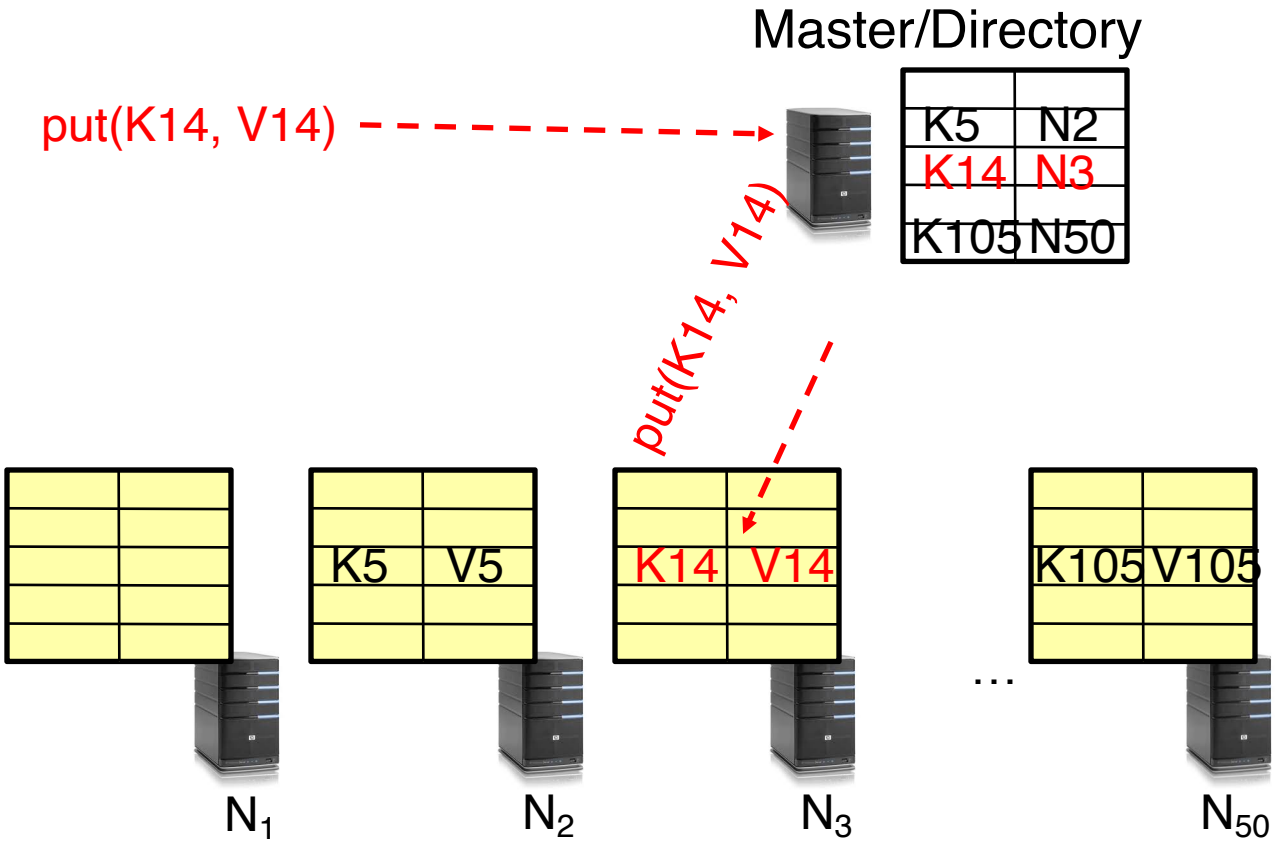
- **Fault Tolerance:** handle machine failures without losing data and without degradation in performance
- **Scalability:**
  - Need to scale to thousands of machines
  - Need to allow easy addition of new machines
- **Consistency:** maintain data consistency in face of node failures and message losses
- **Heterogeneity (if deployed as peer-to-peer systems):**
  - Latency: 1ms to 1000ms
  - Bandwidth: 32Kb/s to 100Mb/s

Key Questions

- **`put(key, value)`: where to store a new (key, value) tuple?**
- **`get(key)`: where is the value associated with a given "key" stored?**
- **And, do the above while providing**
  - Fault Tolerance
  - Scalability
  - Consistency

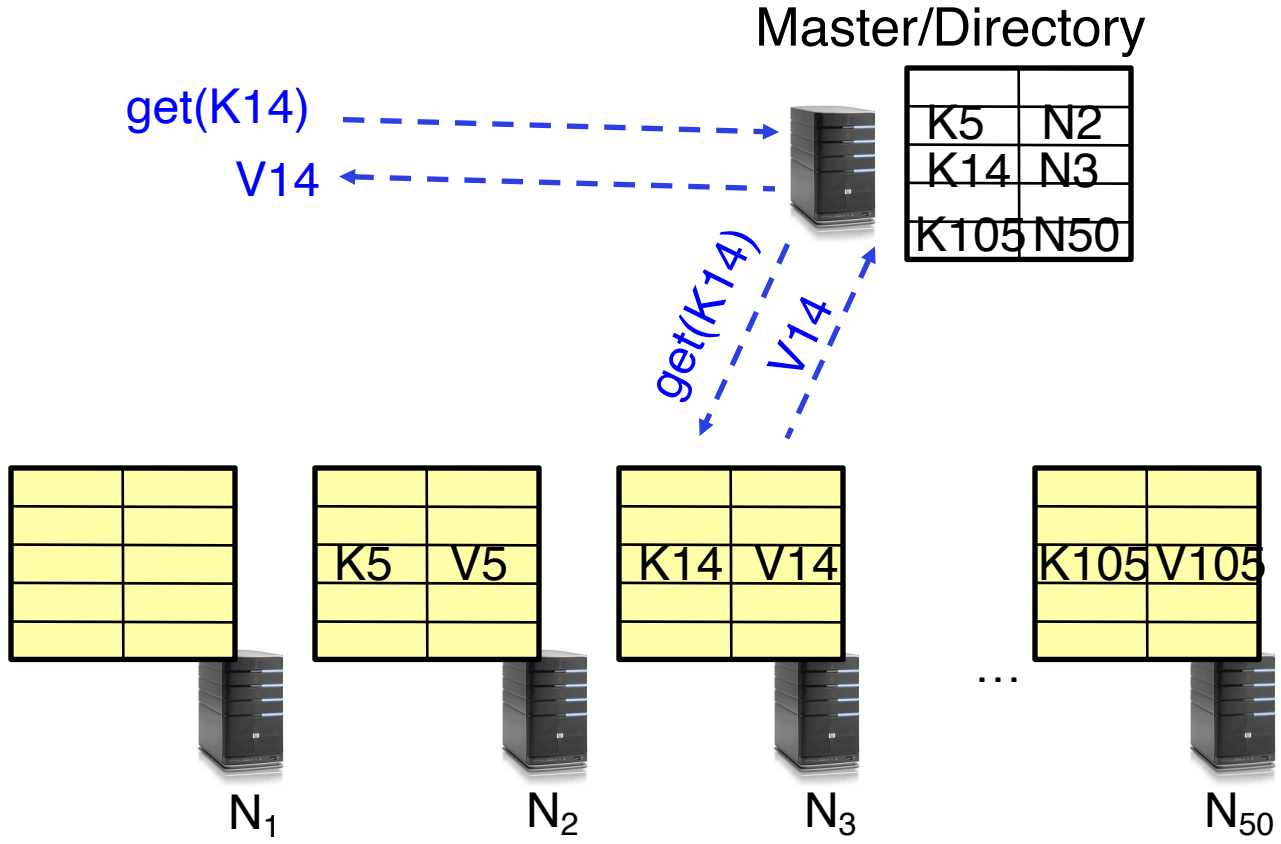
Directory-Based Architecture

- **Have a node maintain the mapping between keys and the machines (nodes) that store the values associated with the keys**



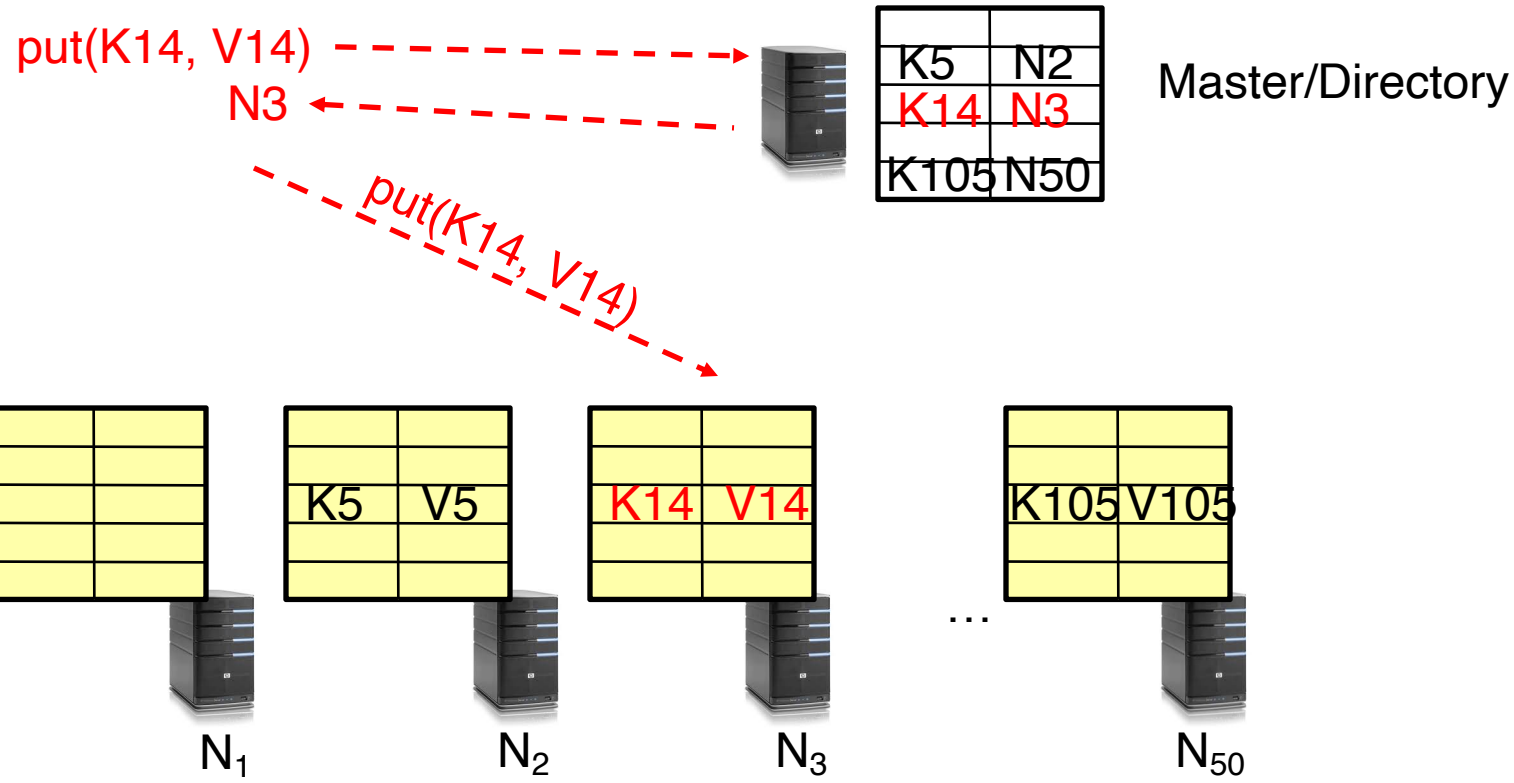
Directory-Based Architecture

- **Have a node maintain the mapping between keys and the machines (nodes) that store the values associated with the keys**



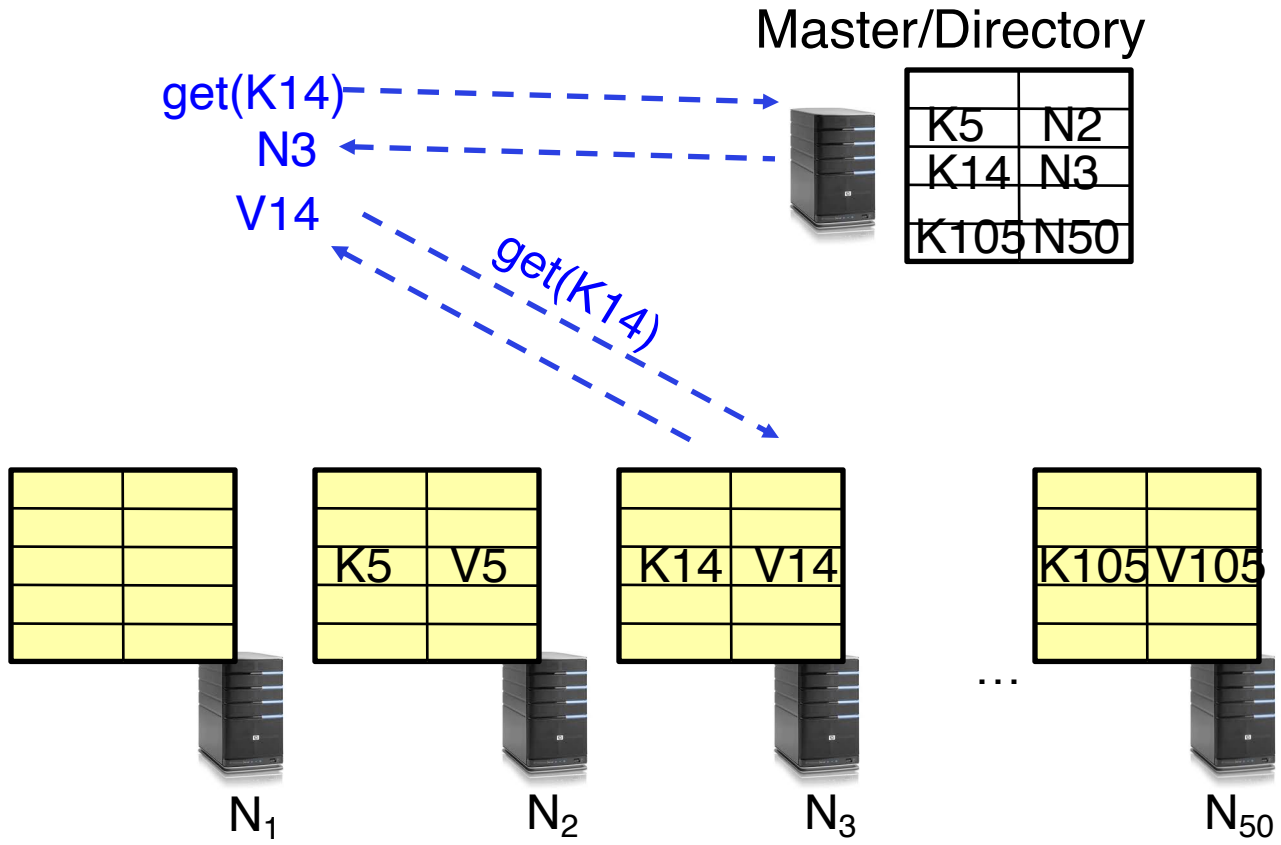
Directory-Based Architecture

- ~~Having the master relay the requests → recursive query~~
- Another method: **iterative query** (this slide)
  - Return node to requester and let requester contact node



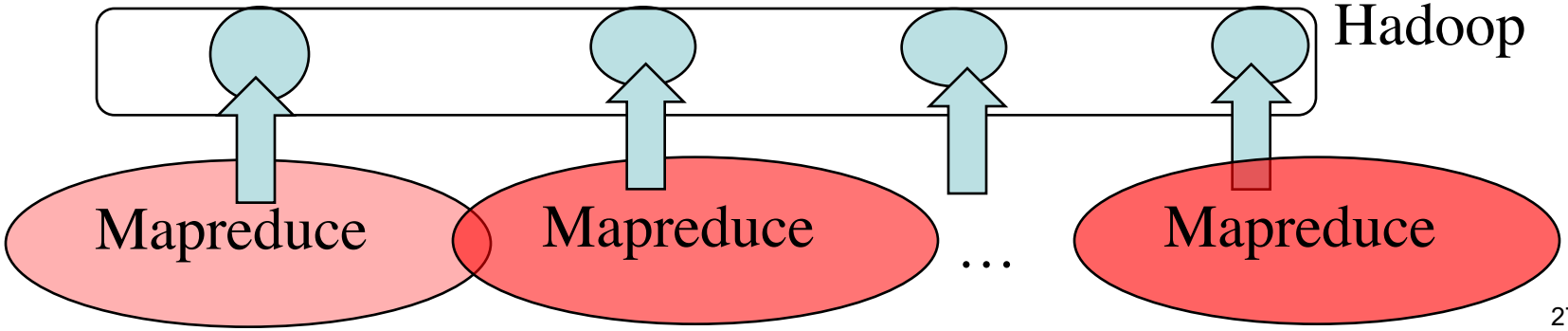
Directory-Based Architecture

- ~~**Having the master relay the requests → recursive query**~~
- **Another method: iterative query**
  - Return node to requester and let requester contact node



Distributed Processing for Indexing and Data Analysis

- Distributed processing driven by need to index and analyze huge amounts of data (i.e., the Web)
- Large numbers of inexpensive servers used rather than larger, more expensive machines
- *MapReduce* is a distributed programming tool
  - Simplify data distribution on a cluster of machines
  - Open source code runs on Hadoop distributed file system
  - Provide fault tolerance
  - But not designed for interactive applications



MapReduce Programming Model

- Data: a set of key-value pairs to model input, intermediate results, and output
  - Initially input data is stored in files
  - stored in Hadoop: distributed file system built on a cluster of machines→ Looks like one machine
- Parallel computation:
  - A set of Map tasks and reduce tasks to access and produce key-value pairs
  - Map Function: (key1, val1) → (key2, val2)
  - Reduce: (key2, [val2 list]) → [val3]

Inspired by LISP Function Programming

- **Lisp *map* function**
  - Input parameters: a function and a set of values
  - This function is applied to each of the values.

  Example:
  - (map 'length '(() (a) (ab) (abc)))

  →(length(()) length(a) length(ab) length(abc)). → (0 1 2 3)

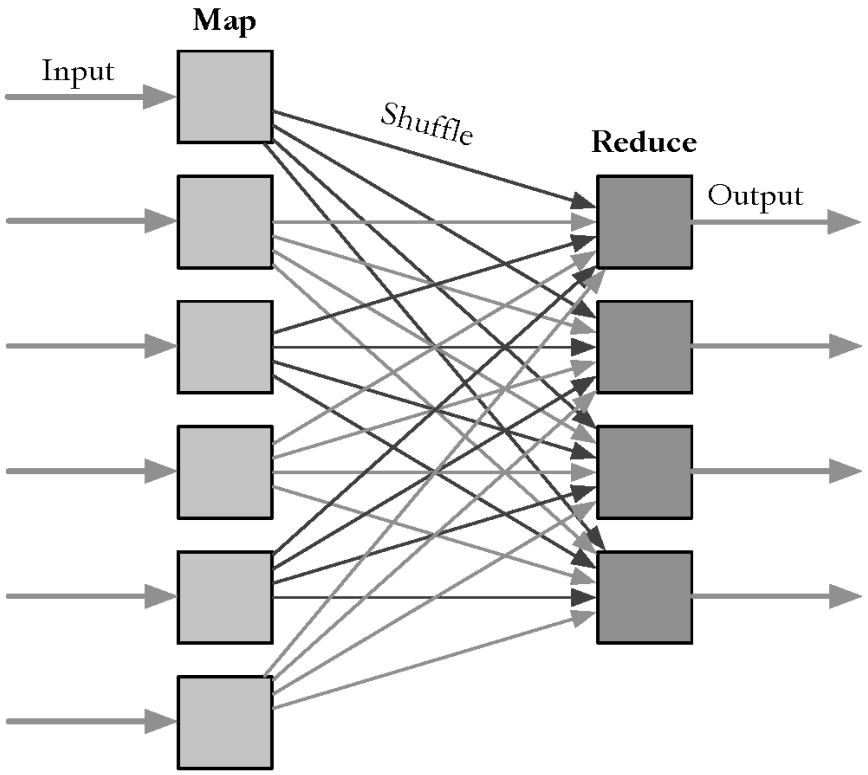
- **Lisp *reduce* function**
  - given a binary function and a set of values.
  - It combines all the values together using the binary function.
  - Example:
    - use the + (add) function to reduce the list (0 1 2 3)
    - (reduce #'+ '(0 1 2 3)) → 6

MapReduce

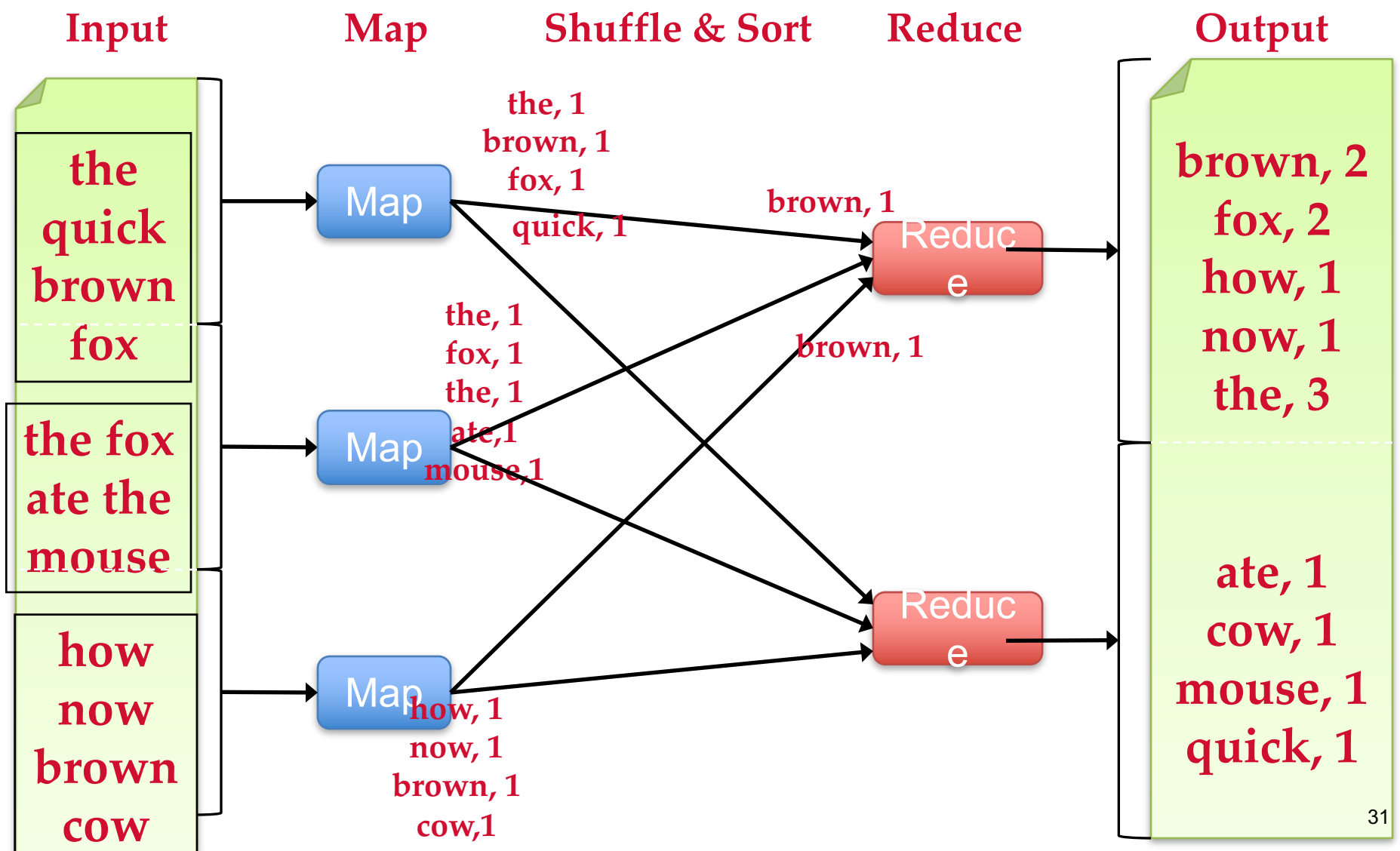
Distributed programming framework that simplifies on data placement and distribution on a cluster of machines

- ***Mapper***
  - Generally, transforms a list of items into another list of items of the same length
- ***Reducer***
  - Transforms a list of items into a single item
  - processes records in batches, where all pairs with the same key are processed at the same time
- ***Shuffle***
  - Uses a hash function so that all pairs with the same key end up the same machine

Suitable for large data mining jobs
Not for interactive jobs



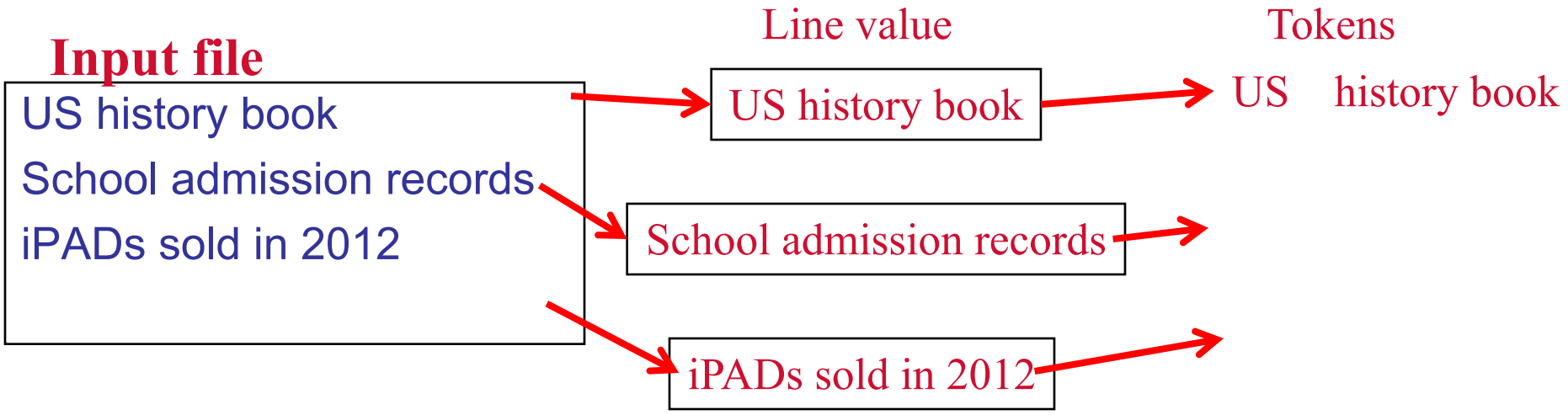
MapReduce to compute document frequency of terms



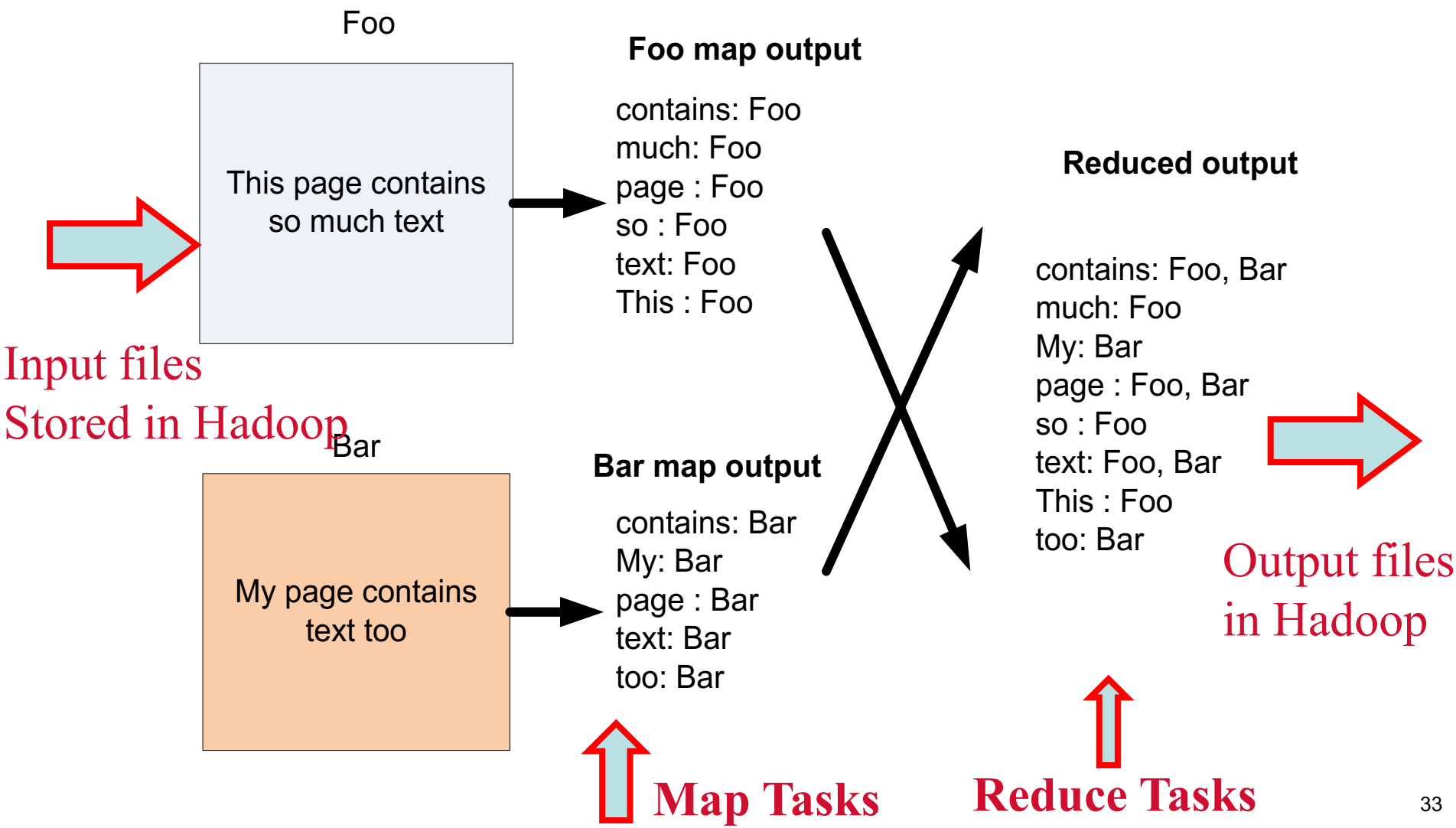
Document Frequency: Input Example

map() gets a key, value

- key - "bytes from the beginning of the line?“
- value - the current line;



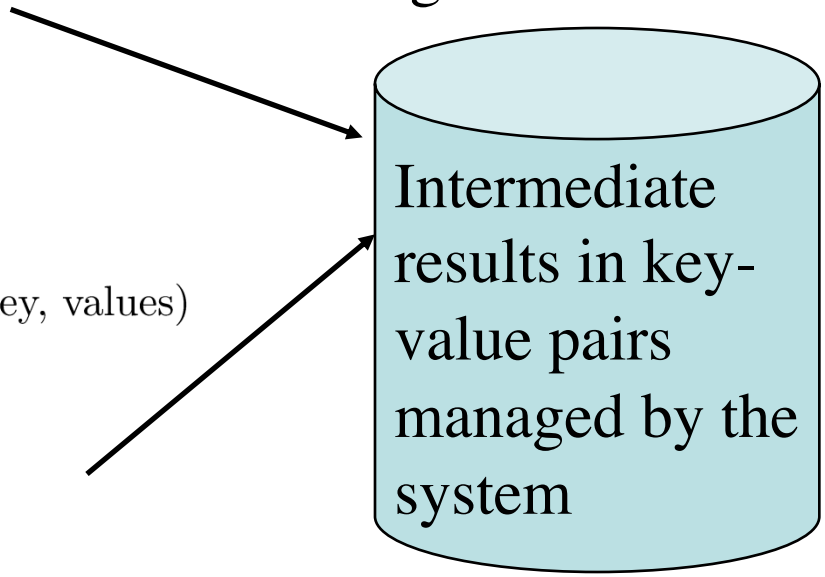
Inverted Indexing with Mapreduce



Pseudo code example for indexing with position information

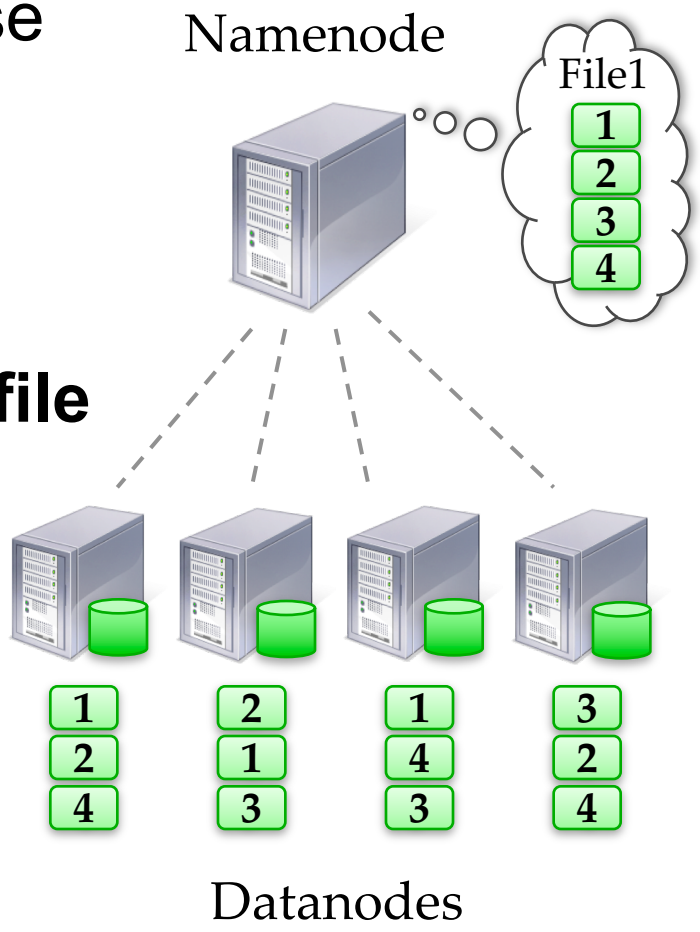
```
procedure MapDocumentsToPostings(input)
    while not input.done() do
        document ← input.next()
        number ← document.number
        position ← 0
        tokens ← Parse(document)
        for each word w in tokens do
            Emit(w, number:position)
            position = position + 1
        end for
    end while
end procedure


  procedure ReducePostingsToLists(key, values)
      word ← key
      WriteWord(word)
      while not input.done() do
          EncodePosting(values.next())
      end while
  end procedure
```

A user writes a small amount of code without worrying about inter-machine management



Hadoop Distributed File System

- **Standard file interface as Linux**
  - Open, seek, read, write, close
- **Files split into 64 MB blocks**
  - Blocks replicated across several datanodes (3)
- **Namenode stores metadata (file names, locations, etc)**
- **Files are append-only. Optimized for large files, sequential reads**
  - Read: use any copy
  - Write: append to 3 replicas



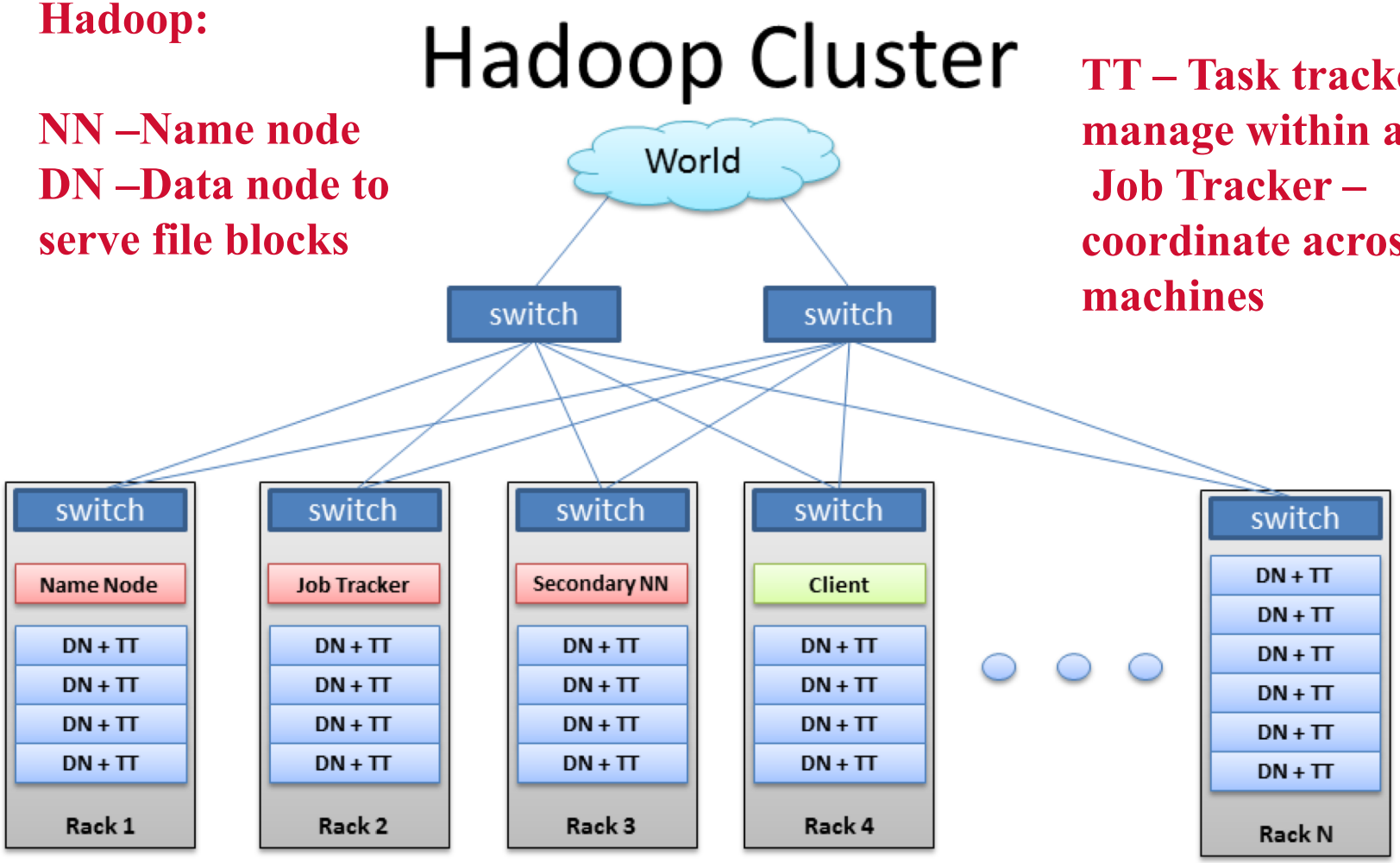
Hadoop Cluster with MapReduce

Daemons for Hadoop:

NN –Name node
DN –Data node to serve file blocks

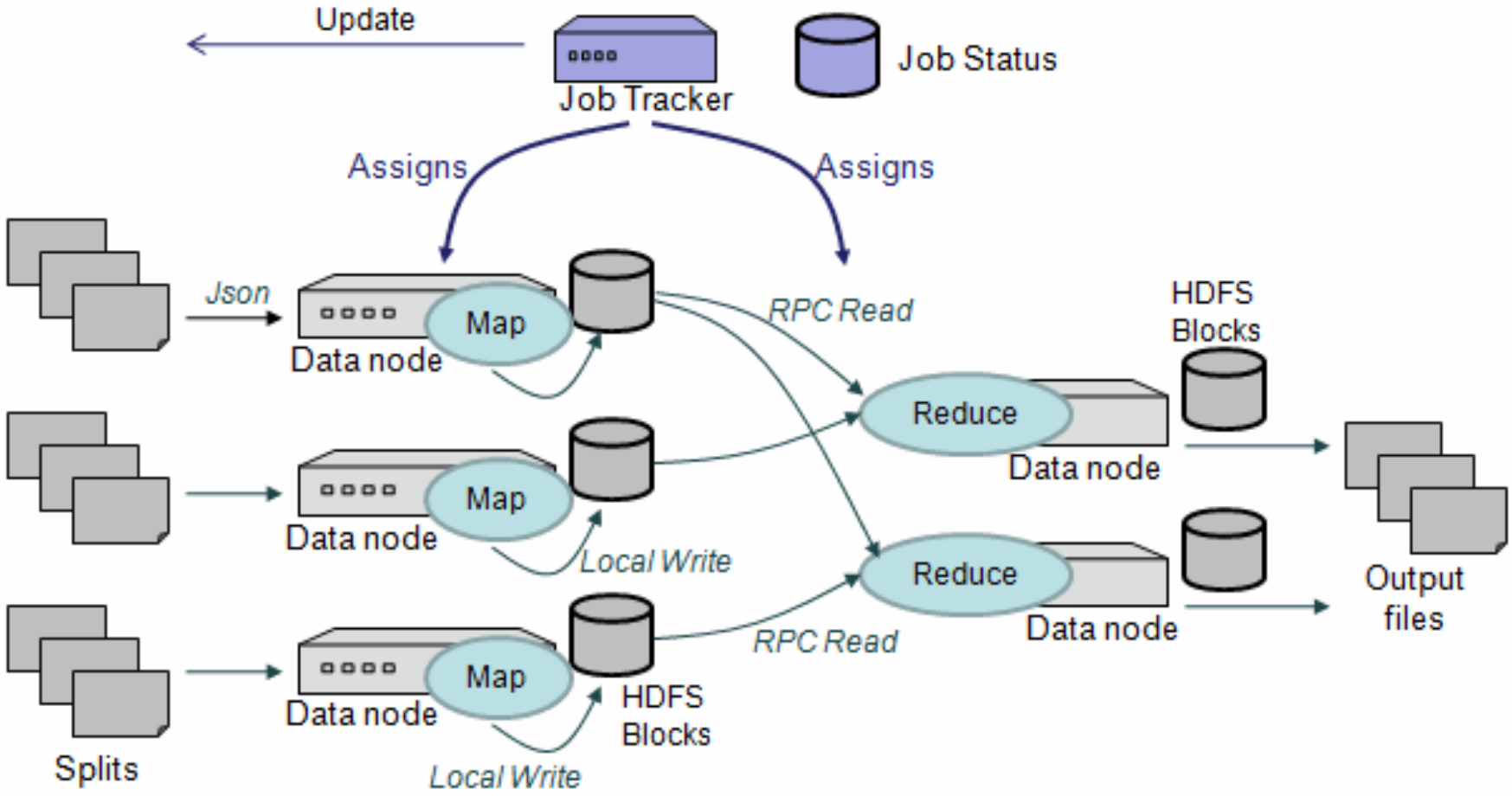
Daemons for MapReduce

TT – Task tracker to manage within a node
Job Tracker – coordinate across machines



BRAD HEDLUND .com

Execute MapReduce on a cluster of machines with Hadoop DFS



Summary

- **Offline incremental data processing**
  - All kinds of text mining and data transformation
    - Indexing, duplicate removal, content classification, spam analysis
  - Combine information from different sources
    - Web pages, entity/knowledge graph, link data, click data, database tables
- **Offline architectures and infrastructure**
  - Flow control for large system components
    - Pipeline, incremental update, 24x7 support
  - Examples of system software for parallel/distributed processing
    - key-value stores, map-reduce