Data Representation and Indexing

Kai Shen

Data Representation

- Data structure and storage format that represents, models, or approximates the original information
- What we've seen/learned so far?
  - Raw byte stream (file)
  - Relational databases with tables, views, keys (references), indexes etc.
  - Key-vale hashtables
  - ⇨ Designed for general purpose
- More complex or customized data representation needed for many big data applications

Data Representation Objectives

- Reduce computation in query and analysis
- Reduce storage and/or communication
- Reduce the statistical/structural complexity
- Data sampling
- … …

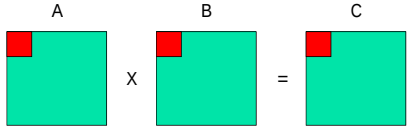
Matrix Representation

- Sparse matrices

41092 41092 1683902
1 1 -0.993907
2 1 -0.110223
1 2 -0.365337
2 2 0.75175
3 3 -0.66616
4 3 0.745809
3 4 0.229777
4 4 0.697537
5 5 0.706766
6 5 -0.707448
… … … …

Matrix Representation

- Block-based data structures for high cache efficiency
- e.g., matrix-matrix multiplication



  - N×N matrix; b×b block
  - Standard operations
    - Each element loaded N times to cache
  - Block operations?
    - Each element loaded N/b times to cache
    - If 3 blocks together fit into the cache

Inverted Indexes

- Mapping from content to its physical location in the dataset
  - Accelerate lookup and search
- In Web keyword search:
  - A search index contains a list of all searchable words, each of which contains a list of documents relevant to the word

Java:	Page #123	Page #157	… …
Sun:	Page #157	Page #468	… …

… …

  - Intersection of document lists for multiple-word queries
    - Fast intersection?

String Matching

- In some applications (e.g., biology), it is useful to query a substring (DNA or protein sequences) in a database
- Suffix-tree
  - http://en.wikipedia.org/wiki/File:Suffix_tree_BANANA.svg
  - Linear time and space construction

Data Size Reduction and Compression

- Big data is large ⇨ reduced size saves storage cost and more importantly network transfer cost
- Deduplication
  - Organize data into fixed-size blocks (e.g., 4KB)
  - Identify duplicate blocks through cryptographic hashing
    - Collision resistance: it is infeasible to find two different messages with the same hash

    http://en.wikipedia.org/wiki/Cryptographic_hash
  - Keep just one physical copy of duplicate blocks and maintain a mapping from logical blocks to the physical location
  - Can be done by the file system and benefit all applications

More on Deduplication

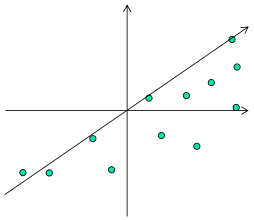
- Inserting or deleting a single byte at the beginning will make all 4KB blocks different from before ⇨ no duplicates?
- Block boundary defined by content, not a fixed size
  - "A Low-bandwidth Network File System", Figure 1

Bloom Filters

- A compressed set representation
  - A bit array, initialized to zero at all bits
  - k hash functions: $h_1()$, $h_2()$, …, $h_k()$ that map element to location in the bit array
  - insert(e) – setting $h_1(e)$, $h_2(e)$, …, $h_k(e)$ to 1 in the bit array
  - lookup(e) – checking whether $h_1(e)$, $h_2(e)$, …, $h_k(e)$ are all 1 in the bit array
- Space saving compared to original set
- Lossy: $h_1(e)$, $h_2(e)$, …, $h_k(e)$ were all set by insertions of other elements
- Allow mathematical analysis on the false positive rate and optimal parameter setting

Complexity Reduction

- Making the data simpler for organization and human exploration
- High-dimensional data
  - Many labels for each user
  - Many attributes for each movie
  - Existence of various DNA sequences in human genome
- Dimension reduction: Principal Component Analysis
  - Finding the direction (in multidimensional space) along which the data spreads out the most



Dimension Reduction Example

- Example on DNA single-nucleotide polymorphisms in human genetic data
  - "PCA-Correlated SNPs for Structure Identification in Worldwide Human Populations" [Paschou et al. 2007]

Data Sampling

- Choose items from a large dataset to form a smaller dataset
  - Smaller dataset is cheaper/quicker to analyze on
  - Yet it preserves the patterns/characteristics of the original dataset to be analyzed
  - e.g., choose a sample set of web pages to test new indexing and searching algorithms
- Random sampling
- Sampling that preserves structures
  - Sample set of web pages that retains references
  - Network/graph sampling

Assignment #4

- Implement word count, matrix vector multiplication, matrix-matrix multiplication, and k-means sampling
- Programing Java (or C/C++) threads
- One easy way of implementation – realize mapreduce control and data transfer in threads
  - Parallel threads for the map() tasks
  - Join the threads when they are done
  - One control thread performs the reduce() tasks

Assignment #4

- Performance can be improved
  - Spread out the work of reduce() tasks
  - Better load balancing
  - … …
- Performance is measured at best speed
  - 24-core or not, you should specify in your TURNIN

Assignment #4

- We will work on a designated 24-core machine
  - Be courteous to each other
- No NFS mounting
  - File system is local on the machine
  - TURNIN doesn't work directly on the machine!!!
  - Data is not backed up ⇨ you should manually back up valuable stuff
  - Don't use too much space
  - Your data is wiped out from the machine after the assignment

Disclaimer

- Preparation of this class was helped by materials in the book “Frontiers in Massive Data Analysis” by the Committee on the Analysis of Massive Data, the Committee on Applied and Theoretical Statistics, the Board on Mathematical Sciences and Their Applications, the Division on Engineering and Physical Sciences, and the National Research Council.