Introduction

• Big Data Definition

1. Big data is a popular term used to describe the exponential growth and availability of data, both structured and unstructured.

2. Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, curation, storage, search, sharing, transfer, analysis, and visualization.

Big Data Acquisition and Management:

1. Acquisition and Recording
2. Information extraction, Cleaning and Annotation
3. Integration, Aggregation, Representation.
4. Query processing, Data modelling, and Analysis

• Big Data Analytics:

1. Analytics: Uses descriptive and predictive models to gain valuable knowledge from data - data analysis. Analytics uses this insight to recommend action or to guide decision making - communication.

2. Big data analytics is the process of examining large amounts of data (big data) which cannot be done by conventional approach, to uncover hidden patterns, unknown correlations and other useful information.

   The technologies associated with big data analytics include NoSQL databases, Hadoop and MapReduce.
Big Data: 5-V’s(3 major-3-v’s)

• **Volume:** Extremely large volume of data; factors contributing to the increase in data volume include transaction-based data stored through the years, unstructured data streaming in from social media, increasing amounts of sensor and machine-to-machine data being collected. In the past, excessive data volume was a storage issue but with decreasing storage costs, other issues emerge, including how to determine relevance within large data volumes and how to use analytics to create value from relevant data. Further values of data explode when it can be linked with other data, thus data integration is a major creator of value: Homeland Security (through analysis of social networks and financial transactions of possible terrorists).

• **Velocity:** the rate at which data arrive and the time in which it must be acted upon.

Data is streaming in at unprecedented speed and must be dealt with in a timely manner. RFID tags (Radio-frequency identification for the purposes of automatically identifying and tracking tags attached to objects for wireless transmission of data), sensors and smart metering are driving the need to deal with torrents of data in near-real time. Reacting quickly enough to deal with data velocity is a challenge for most organizations.

• **Variety:** heterogeneity of data types, representation, and semantic interpretation

Data today comes in all types of formats; structured, numeric data in traditional databases, information created from line-of-business applications, unstructured text documents, web data (structured and unstructures), social network data- Graph data, email, video, audio, stock ticker data and financial transactions. Managing, merging and governing different varieties of data is something many organizations still grapple with.

Other important characteristics:

• **Variability:** In addition to the increasing velocities and varieties of data, data flows can be highly inconsistent with periodic peaks.
Is something trending in social media? Daily, seasonal and event-triggered peak data loads can be challenging to manage. Even more so with unstructured data involved.

- **Veracity:** Data needs cleaning because not all of it may not be correct.

- **Complexity:** Today’s data comes from multiple sources. And it is still an undertaking to link, match, cleanse and transform data across systems. However, it is necessary to connect and correlate relationships, hierarchies and multiple data linkages or your data can quickly spiral out of control.

- **Privacy**

- **Usability**
Examples of BIG DATA

1. RFID (radio frequency ID) systems generate up to 1,000 times the data of conventional bar code systems.

2. In just four hours on “black Friday” 2012, Walmart handled 10 million cash register transactions almost 5,000 items per second.

3. United Parcel Service receives on average 39.5 million tracking requests from customers per day.

4. VISA processes more than 172,800,000 card transactions each day. 500 million tweets are sent per day. That’s more than 5,700 tweets per second.

5. Facebook has more than 1.15 billion active users generating social interaction data.

6. More than 5 billion people are calling, texting, tweeting and browsing websites on mobile phones.

7. Affordable Health Care: User information, Insurance information, many others.

8. Webdata: Search engines

9. The Sloan Digital Sky Survey [SDSS2008] has today become a central resource for astronomers the world over

10. Biology: with the advent of Next Generation Sequencing, the size and number of experimental data sets available is increasing exponentially.

11. In 2010 enterprises and users stored more than 13 exabytes of new data:

12. Even declarative languages that exploit mapReduce, such as Pig Latin, are at a rather low level when it comes to complex analysis tasks
Big Data Acquisition and Management

• **Acquisition and Recording**
  
  Acquire (e.g., through sensors), filter out unnecessary data while keeping the needed data; automatically generate meta data to describe the data being recorded. Because of the huge amount this has to be automated and stored only after filtering.

• **Information extraction, Cleaning and Annotation**
  
  Information extraction process pulls out the required information from the underlying sources and expresses it in a structured form suitable for analysis.

• **Integration, Aggregation, Representation.**
  
  For large scale analysis this has to happen in an automated manner.

• **Query processing, Data modelling, and Analysis**
  
  Querying and Mining big data is fundamentally different from traditional statistical analysis on small data. Big data is often, noisy, dynamic, heterogeneous, untrustworthy. Real life Real-life medical records have errors, heterogeneous and distributed across multiple systems.
Data Management systems for Big Data

• Comparison with relational database systems:
  1. Needs more flexible data model
  2. Needs to handle internet scale data
  3. Model not optimized using distributed and massively parallel processing with possible failures.

• SQL is not the right choice; different data models

• Queries can be batch processing (mapReduce), online processing (Google Big Query)
NoSQL Databases

• Scaling is easier
• Schema-free design
• New datatypes dynamically created on the fly.
• Easy Replications
• Maps into hugely Parallel processing easily (key-value pairs similar to data structure used in Map/Reduce.)
NoSQL Databases

● Definition:
  1. Less ACID properties
  2. NoSQL is a movement (no standard yet) promoting a loosely de-
     fined class of non-relational data stores.

● Horizontally Scaling (scale out): Add more nodes.
  as opposed to vertically Scaling in traditional approach (scale Up):
  Add resorces (memory, cpu, etc.) to a single node.

● Auto Sharding
  A NoSQL database automatically spreads data (share nothing, hori-
  zontal partitioning) across servers, without requiring applications to
  participate.

● Users
  Major extremely-used sites and services such as Facebook, LinkedIn,
  Twitter, etc. are known to use them extensively to achieve scalability
  and agility.

● Systems: Oracle, Google Big Table, MongoDB, Appache-Cassandra,
  HBase, Neo4j
NoSQL Database Models

- Key value stores
- Column family stores
- Document Stores
- Graph Stores
Key Value Store

• Think about Map-Reduce- Key-Value pairs.

• Counting key word- Example
  Database has key-defined: document ID, Value: Document (long string), Application is parsing the document.

• Simplest NoSQL database: every single item in a file is stored as an attribute name (or ”key”) together with its value (uninterpreted byte arrays).

• There is no structure nor relation. An application can state a key (e.g. the-answer-to-life) and provide a matching value (e.g. 42) which can later be retrieved the same way by supplying the key (like dictionary).

• Thus, data can be retrieved (searched) only by key. Due to this simple structure key-value store is completely schema free.

• Relationship between data must be handled in application logic.

• New values can be added at runtime without conflicting any other stored data; does not matter duplicate or not.

• Oracle KV( Key-Value) Store, Cassandra
• They are extremely efficient and usually easily scalable.
Oracle KV-Store

- User-defined keys (formatted as Strings) to opaque data items
- Major Key, Minor Key
- Major Key path, Minor key path defined by the applications.
- A key is the concatenation of a major key path and a minor key path.
- All records sharing a major key path are co-located to achieve data locality. Within the collection of of major key path data is organized by minor key payh.
- For example, an application storing user profiles might use the profile-name as a Major Key Path
- Several Minor Key Paths for different components of that profile such as email address, name, phone number, etc.
- Oracle NoSQL Database hashes keys to provide good distribution over a collection of computers that provide storage for the database.
- There are no restrictions on the size or structure of the value field.
Oracle Shards

- A database shard is a horizontal partition of data in a database or search engine.
- Each individual partition is referred to as a shard.
- Each shard is held on a separate database server instance, to spread load.
- Sharding goes beyond horizontal partitioning of relations in that it handles replications.
Oracle NoSQL Examples

- Provides some table structure, Key corresponds to a row of the table.

- *** Enter into table creation mode
  ```
  table create -name myProducts
  ```
  # Now add the fields
  ```
  add-field -type STRING -name productName
  add-field -type STRING -name productType
  add-field -type STRING -name productClass
  add-field -type ENUM -name color -enum-values blue,green,red
  add-field -type ENUM -name size -enum-values small,medium,large
  add-field -type INTEGER -name inventoryCount
  ```
  # A primary key must be defined for every table
  ```
  primary-key -field productName -field productType -field productClass
  ```
  # Exit table creation mode
  ```
  exit
  ```
  # Add the table to the store. Use the -wait flag to force the script to wait for the plan to complete before doing anything else.
  ```
  plan add-table -name myProducts -wait
  ```

- full primary key fields: productName, productType, and productClass.

- Partial set of fields can be used but the order matters.
  ```
  productName
  ```
productName, productType are valid but

productType and productClass is not.

Oracle uses hash function in a particular way.

• Partial key fields useful for querying, deleting, etc. for a set of rows at a time.

• Key fields can be of types
  . Integer
  . Long
  . Float
  . Double
  . String
  . Enum

• Shard Keys:

  Subset of the key fields
  Primary key fields meaningful in terms of shard storage.

• Rows which contain the same values for all the shard key fields are guaranteed to be stored on the same shard.

  shard-key -field productType -field productName
Instances of the data

• ## Enter into table creation modetable create -name
## Now add the fieldsadd-field -type STRING -name item type
add-field -type STRING -name item category
add-field -type STRING -name item class
add-field -type STRING -name item color
add-field -type STRING -name item sizeadd-field -type
add-field -type INTEGER -name inventory count
primary-key -field item type -field item category -field
-field item color -field item size
shard-key -field item type -field item category -field
## Exit table creation modeexit
With tables containing data like this:

. Row 1:
itemType: Hats
itemCategory: baseball
itemClass: longbillitemColor: red
itemSize: small
price: 12.07
inventoryCount: 127
. Row 2:
itemType: Hats
itemCategory: baseball
itemClass: longbill
itemColor: red
itemSize: medium
price: 13.07
inventoryCount: 201

Row 3:
itemType: Hats
itemCategory: baseball
itemClass: longbill
itemColor: red
itemSize: large
price: 14.07
inventoryCount: 39
Child Tables

- `table create -name userInfo## Now add the fields`
  - `add-field -type STRING -name surname`
  - `add-field -type STRING -name gender`
  - `add-field -type STRING -name street`
  - `add-field -type STRING -name state`
  - `add-field -type STRING -name userPhone`
  - `add-field -type BINARY -name publickey1`
  - `primary-key -field surname -field familiarName`
  - `shard-key -field surname## Exit table creation mode`
  - `exit`

- `### Must add the parent table before we add the child table create -name userInfo.largeProps -type BINARY -name voiceGreeting`
  - `add-field -type BINARY -name imageFile`
  - `primary-key -field propType`
  - `exit`
  - `plan add-table -name userInfo.largeProps -wait`

- `Avoid retrieving the big data values whenever you retrieve a row by splitting the image data and voice greeting into a child table.`
Avro Schema/JASON

{"namespace": "example.avro",
 "type": "record",
 "name": "User",
 "fields": [
   {"name": "name", "type": "string"},
   {"name": "favorite_number", "type": ["int", "null"]},
   {"name": "favorite_color", "type": ["string", "null"]}
 ]
}
COLUMN FAMILY STORES (based on Google Big Table)

- Allowing slightly more structure to the key-value based approach. Collections of one or more key value pairs for a corresponding row/key.
- In other words, it is a tuple (pair) that consists of a key-value pair, where the key corresponds to a row which maps to a value that is a set of columns with column name and value.
- Each column is a tuple (triplet) consisting of a column name, a value, and a timestamp.
- Column-based NoSQL databases are two dimensional arrays whereby each key (i.e. row / record) has one or more key / value pairs attached to it.
- Column families have a schemeless nature so that each of their "row"s can contain a different number of columns, and even different column names could be in each row. So, they are a very different concept than the rows in relational database management system (RDBMS)s.

UserList={
    Cath: {
        username: {firstname:.Cath., lastname:.Yoon.}
    }
}
Where "Cath" and "Terry" are row keys; "username", "account", and "address" are key-value pairs.

- In a relational database table, this data would be grouped together within a table with other non-related data.
- These databases are commonly used when simple key / value pairs are not enough, and storing very large numbers of records with very large numbers of information is a must. DBMS implementing column-based, schema-less models can scale extremely well.
- Big Table is designed for storing items such as billions of URLs, with many versions per page; over 100 TB of satellite image data; hundreds of millions of users; and performing thousands of queries a second.
- Bigtable is designed with semi-structured data.
storage in mind.

- Indexed by a row key, column key, and a timestamp
- Google Big Table on GFS (Google File System)
- Open source version of Google Big Table is HBase: created by the Apache project on top of the Hadoop core.
- MAP: it is an associative array; a data structure that allows one to look up a value to a corresponding key quickly. Bigtable is a collection of (key, value) pairs where the key identifies a row and the value is the set of columns.

- \((row, column, timestamp) \rightarrow \text{String}\)
- Right side ”string” is the value.
- Example: Want to keep a copy of a large collection of web pages and related informations:
  - Create a table called webTable
  - URL as row KEY
  - Big Table keeps data in lexicographic order by row key
  - Store URL in reverse order (com.cnn.www) so that pages in the same domain are grouped together.
Various aspects of the web page as COLUMN NAMES

Contents of the web pages in the column "contents" under the time stamp they were fetched.

Column family: column keys are grouped into sets called column families.

Example for the web table: column family is LANGUAGE column key is English in this particular case there is only one column key for the column family.

A column key is named using the syntax familyname: qualifier. for example a column key is LANGUAGE: English.

Thus a column family must be created before data can be stored under any column key.

A column family is the basic unit for access control.

All data stored in a column family is usually of the same type (all data of a column family compressed together)

Multidimensional:

```
edu.rutgers.cs" : { // row
    "users" : { // column family
        "watrous": "Donald", // column
```
"hedrick": "Charles", // column
"pxk": "Paul" // column }
"sysinfo": { // another column family
"": "SunOS 5.8" // column (null name)

- **Column and Column Family:**

<table>
<thead>
<tr>
<th>row keys</th>
<th>column family</th>
<th>column family</th>
<th>column family</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.aaa</td>
<td>&quot;language:&quot;</td>
<td>&quot;contents&quot;</td>
<td>anchor.cnnsi.com</td>
</tr>
<tr>
<td>com.cnn.www</td>
<td>ENG</td>
<td>&lt;DOCTYPE.html</td>
<td>anchor.mylook.ca</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;DOCTYPE html</td>
<td>&quot;CNN&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;CNN.com&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2:

- Store columns of data together, instead of rows.
- Similar to key, value stores, values cannot be interpreted by the system;
- Columns and rows can be added flexibly at run time. Column family have to be predefined often times making it less flexible than key, value store.
- Many projects in google use this model: Web indexing, Google earth.
- Big Table can be used for MapReduce: as an input source and as an output target.
- Google Big Query:
  SQL like structure:
All BigQuery queries are SELECT statements of this form:

```sql
SELECT expr1 [[AS] alias1] [, expr2 [[AS] alias2], ...]
[agg_function(expr3) WITHIN expr4]
[FROM [(FLATTEN(table_name1|(subselect1)]) [, table
([(INNER]|LEFT OUTER) JOIN table_2|(subselect2)
ON join_condition_1 [... AND join_condition_N
[WHERE condition]
[GROUP BY field1|alias1 [, field2|alias2, ...]]]
[HAVING condition]
[ORDER BY field1|alias1 [DESC|ASC] [, field2|ali
[LIMIT n]
;

SELECT
/* Replace underscores in the title with spaces */
REGEXP_REPLACE(title, r'_', ' ') AS regexp_title,
FROM
(SELECT title, SUM(views) as views
FROM [bigquery-samples:wikimedia_pageviews.201201]
WHERE
NOT title CONTAINS ':'
AND wikimedia_project='wp'
AND language='en'
/* Match titles that start with 'G', */
/* end with 'e', and contain two 'o's */
AND REGEXP_MATCH(title, r'^G.*o.*o.*e$')
```
GROUP BY
title
ORDER BY
views DESC
LIMIT 100)

Note: Keywords are not case-sensitive. In this document, keywords such as SELECT are capitalized for illustration purposes.
DOCUMENT STORES

- This is not document databases (sort of misleading terminology)
- Takes the data you want to store and aggregates it into documents using the JSON format (JavaScript Object Notation).
- This is not document databases (sort of misleading terminology)
- Takes the data you want to store and aggregates it into documents using the JSON format (JavaScript Object Notation).
- is an open standard format that uses human-readable text to transmit data objects consisting of attribute.value pairs. JSON is a syntax for storing and exchanging data. It is self describing like XML.
- JSON is an easier to use alternative to XML. JSON uses JavaScript, but the JSON format is text only, just like XML. JSON is a syntax for storing and exchanging data. It is self describing like XML.
- JSON is an easier to use alternative to XML.

JSON Example
pair each key with complex data structure known as document. Documents: many key-value pairs

For example, take all the data stored in a row that spans 20 tables of a relational database and aggregate it into a single document/object.

Aggregating this information may lead to duplication of information, but since storage is
no longer cost prohibitive, the resulting data model flexibility, ease of efficiently distributing the resulting documents and read and write performance improvements make it an easy trade-off for web-based applications. Remember, NO ACID.

- Idea is to have almost no(?) schema
  (adapts to big data requirements: "Variety")
Example:

```json
{
  "ID": 1,
  "FIRST": "FRANK",
  "LAST": "Weigel",
  "Zip": "94040",
  "CITY": "MV",
  "STATE": "CA"
}
```

<table>
<thead>
<tr>
<th>Key</th>
<th>First</th>
<th>Last</th>
<th>Zip_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Frank</td>
<td>Weigel</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Ali</td>
<td>Dodson</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Mark</td>
<td>Azad</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Steve</td>
<td>Yen</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Zip_id</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DEN</td>
<td>CO</td>
<td>30303</td>
</tr>
<tr>
<td>2</td>
<td>MV</td>
<td>CA</td>
<td>94040</td>
</tr>
<tr>
<td>3</td>
<td>CHI</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>4</td>
<td>NY</td>
<td>NY</td>
<td>10010</td>
</tr>
</tbody>
</table>

No schema, entire relational schema is transformed into document

Figure 3:

- A column family is like a table. A row is like a table row. Columns are sort of like database columns, except that they can be defined on the fly, so you may have a very sparsely-populated table in some cases, or you may have different columns populated in each row.
GRAPH STORES

• As opposed to key, value and Column family, graph databases are specialized on efficient management of linked data. Social Networks.

• Based on directed and multi relational property graphs. Nodes and edges consists of objects with embedded key value pairs. Range of keys and values can be defined in a schema.