DATA ANALYTICS

Comp 624

Daw Pwint Phyu Khine

Big Data

- Big data is typically broken down by three characteristics:

✓ **Volume:** How much data

✓ **Velocity:** How fast that data is processed

- batch
- real time
- near real time
- streaming

✓ **Variety:** The various types of data

- structured, unstructured (semi-structure)

<<<Veracity>>>,

<<<Value>>>

Data

Data in motion

- Data in motion would be used if a company is able to analyze the quality of its products during the manufacturing process to avoid costly errors.

Data At rest

- Data at rest would be used by a business analyst to better understand customers' current buying patterns based on all aspects of the customer relationship, including sales, social media data, and customer service interactions.

Big Data Management Cycle

- It is necessary to identify the right amount and types of data that can be analyzed to impact business outcomes.
- Big data incorporates all data, including structured data and unstructured data from e-mail, social media, text streams, and more.

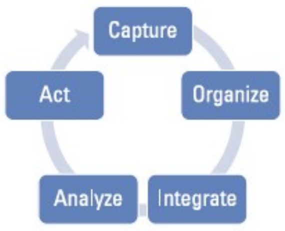
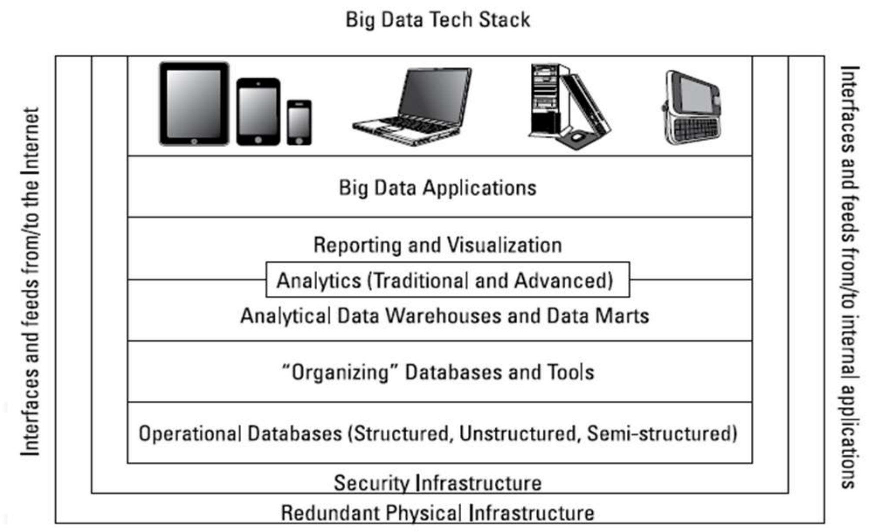


Figure 1-1: The cycle of big data management.

Big Data Architecture


Big Data Tech Stack
Interfaces and feeds from/to the Internet
Interfaces and feeds from/to internal applications
Big Data Applications
Reporting and Visualization
Analytics (Traditional and Advanced)
Analytical Data Warehouses and Data Marts
"Organizing" Databases and Tools
Operational Databases (Structured, Unstructured, Semi-structured)
Security Infrastructure
Redundant Physical Infrastructure

How Data are generated or captured

- Two factors are new in the big data world:

✓ Some sources of big data are actually new like the data generated from sensors, smartphone, and tablets.

✓ Previously produced data hadn't been captured or stored and analyzed in a usable way.

Understanding Data

- Different Types of Data
  - Structured Data - Traditional types of data
  - Unstructured Data – Image, Voice, Audio, Video
  - Semi-structured Data – Time series, Temporal, Spatial

Structured Data

- The sources of structured data are divided into two categories:
  - ✓ **Computer- or machine-generated:** Machine-generated data generally refers to data that is created by a machine without human intervention.
    - Sensor Data -radio frequency ID (RFID) tags, smart meters, medical devices, and Global Positioning System (GPS) data
    - Web log Data
    - Point of Sale
    - Financial Data
  - ✓ **Human-generated:** This is data that humans, in interaction with computers, supply.
    - Input Data e.g. name, age, non-free-form response
    - Click-stream data
    - Gaming data

Relational (Structured Data)

- In a relational model, the data is stored in a table.
- This database would contain a *schema* — that is, a structural representation of what is in the database.
  - For example, in a relational database, the schema defines the tables, the fields in the tables, and the relationships between the two.
- The data is stored in columns, one each for each specific attribute.
- The data is also stored in the rows.

```
Select CustomerID, State, Gender, Product from
            "demographic table", "product table" where
            Product= XXYY
```

Schema
Relationship
SQL(Structured Query Language)

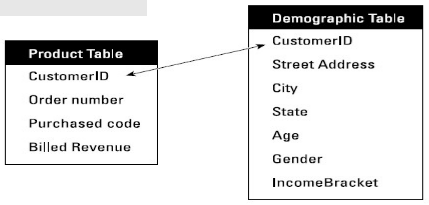


Figure 2-1: The relationships between tables.

Unstructured Data

- *Unstructured data* is data that does not follow a specified format.
- If 20 percent of the data available to enterprises is structured data, the other 80 percent is unstructured.
- Unstructured data is really most of the data that you will encounter.

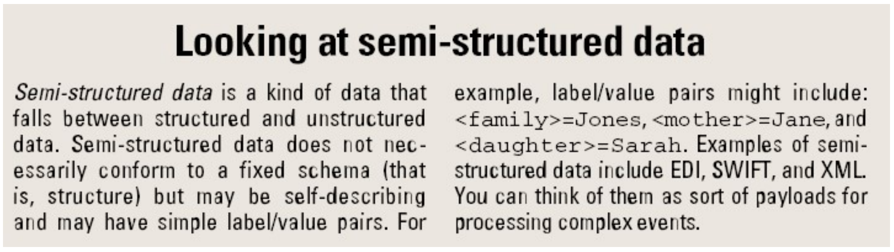
Sources of Unstructured Data

Machine-generated unstructured data

- **Satellite Data:** weather data or the data that the government captures in its satellite surveillance imagery e.g. Google earth.
- **Scientific data:** This includes seismic imagery, atmospheric data, and high energy physics.
- **Photographs and video:** This includes security, surveillance, and traffic video.
- **Radar or sonar data:** This includes vehicular, meteorological, and oceanographic seismic profiles.

- Human-generated unstructured data:
  - ✓ **Text internal to your company:** Think of all the text within documents, logs, survey results, and e-mails.
  - ✓ **Social media data:** Generated from the social media platforms such as YouTube, Facebook, Twitter, LinkedIn, and Flickr.
  - ✓ **Mobile data:** This includes data such as text messages and location information.
  - ✓ **Website content:** This comes from any site delivering unstructured content, like YouTube, Flickr, or Instagram.

Semi-structured Data



Looking at semi-structured data

Semi-structured data is a kind of data that falls between structured and unstructured data. Semi-structured data does not necessarily conform to a fixed schema (that is, structure) but may be self-describing and may have simple label/value pairs. For example, label/value pairs might include: `<family>=Jones`, `<mother>=Jane`, and `<daughter>=Sarah`. Examples of semi-structured data include EDI, SWIFT, and XML. You can think of them as sort of payloads for processing complex events.

Looking at Real-Time and Non-Real-Time Requirements

- Big data approaches will help keep things in balance so we don't go over the edge as the volume, variety, and velocity of data changes
- In general, this real-time approach is most relevant when the answer to a problem is time sensitive and business critical.
  - This may be related to a threat to something important like detecting the performance of hospital equipment or anticipating a potential intrusion risk.
- Sometimes streaming data is coming in really fast and does not include a wide variety of sources, sometimes a wide variety exists, and sometimes it is a combination of the two.

Considering the Real time

Regarding a system's capability to ingest data, process it, and analyze it in real time:

✓ **Low latency:** Latency is the amount of time lag that enables a service to execute in an environment.

- Some applications require less latency, which means that they need to respond in real time.
- A real-time stream is going to require low latency.
- So you need to be thinking about compute power as well as network constraints.

✓ **Scalability:** Scalability is the capability to sustain a certain level of performance even under increasing loads.

✓ **Versatility:** The system must support both structured and unstructured data streams.

✓ **Native format:** Use the data in its native form. Transformation takes time and money.

The capability to use the idea of processing complex interactions in the data that trigger events may be transformational.

Connector -> from twitter(X) to Facebook

Meta Data -> Data about Data

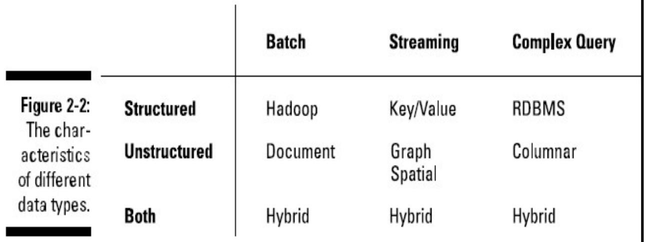


Figure 2-2: The characteristics of different data types.

	Batch	Streaming	Complex Query
Structured	Hadoop	Key/Value	RDBMS
Unstructured	Document	Graph Spatial	Columnar
Both	Hybrid	Hybrid	Hybrid

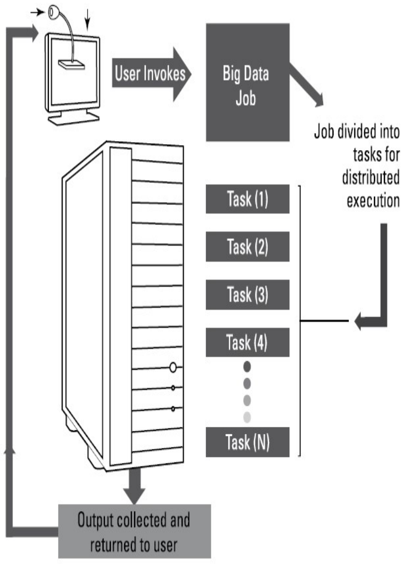
Why we need distributed computing for big data?

- Not all problems require distributed computing.
- One of the perennial problems with managing data — especially large quantities of data — has been the impact of latency.
  - ***Latency* is the delay within a system based on delays in execution of a task.**
- Distributed computing and parallel processing techniques can make a significant difference in the latency experience.
- Many big data applications are dependent on low latency because of the big data requirements for speed and the volume and variety of the data.

Distributed work

- Distribute components of your big data service across a series of nodes.
- In distributed computing, a *node* is an element contained within a cluster of systems or within a rack.
- A node typically includes CPU, memory, and some kind of disk
- Or a blade CPU and memory that rely on nearby storage within a rack.
- Within a big data environment, these nodes are typically clustered together to provide scale.

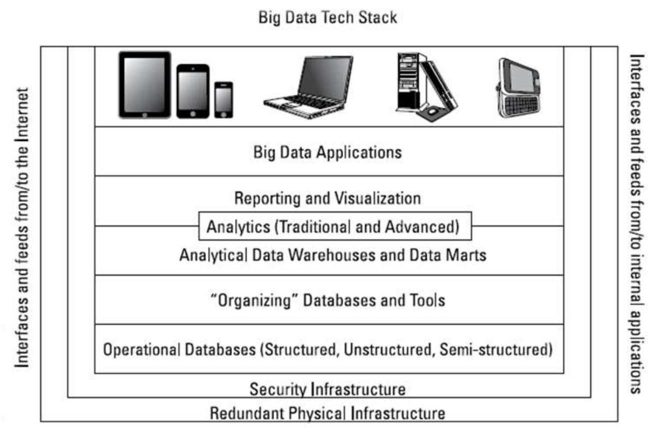
Figure 3-3: An evolution of distributed computing relies on jobs being divided for distributed execution of tasks.


User Invokes
Big Data Job
Job divided into tasks for distributed execution
Task (1)
Task (2)
Task (3)
Task (4)
Task (N)
Output collected and returned to user

Usage of Distributed Computing in Big Data

- To accommodate the growth, an organization simply adds more nodes into a cluster so that it can scale out to accommodate growing requirements.
- it is important to be able to send part of the big data analysis to different physical environments.
  - Where you send these tasks and how you manage them makes the difference between success and failure.
- The closer together the distributions of functions are, the faster they can execute.
- Therefore, scalability is the lynchpin of making big data operate successfully.
- Big data requires the inclusion of fast networks and inexpensive clusters of hardware that can be combined in racks to increase performance by dynamic scaling and load balancing.
- The design and implementations of MapReduce are excellent examples of how distributed computing can make big data operationally visible and affordable.

Implementation of Big Data Architecture


Big Data Tech Stack
Big Data Applications
Reporting and Visualization
Analytics (Traditional and Advanced)
Analytical Data Warehouses and Data Marts
"Organizing" Databases and Tools
Operational Databases (Structured, Unstructured, Semi-structured)
Security Infrastructure
Redundant Physical Infrastructure
Interfaces and feeds from/to the Internet
Interfaces and feeds from/to internal applications

Layer 0: Redundant Physical Infrastructure

- At the lowest level of the stack is the physical infrastructure — the hardware, network, and so on.
- **Performance:**How responsive do you need the system to be? Performance, also called *latency,* is often measured end to end, based on a single transaction or query request.
- **Availability:** Do you need a 100 percent uptime guarantee of service? How long can your business wait in the case of a service interruption or failure?
- **Scalability:** How big does your infrastructure need to be? How much disk space and computing power are needed? Decide what you need and then add a little more scale for unexpected challenges.
- **Flexibility:** How quickly can you add more resources to the infrastructure? How quickly can your infrastructure recover from failures?
- **Cost:** What can you afford?

As big data is all about high-velocity, high-volume, and high-data variety, the physical infrastructure will literally "make or break" the implementation.

Layer 1: Security Infrastructure

- **Data access:** Most core data storage platforms have rigorous security schemes and are often augmented with a federated identity capability, providing appropriate access across the many layers of the architecture.
- **Application access:** Application access to data is also relatively straightforward from a technical perspective. Most application programming interfaces (APIs) offer protection from unauthorized usage or access.
- **Data Encryption**: Encrypting and decrypting data really stresses the systems' resources. With the volume, velocity, and varieties associated with big data, this problem is exacerbated. Try to identify the data elements requiring different level of security and to encrypt only the necessary items.
- **Threat detection:** The inclusion of mobile devices and social networks exponentially increases both the amount of data and the opportunities for security threats. It is therefore important that organizations take a multiperimeter approach to security.

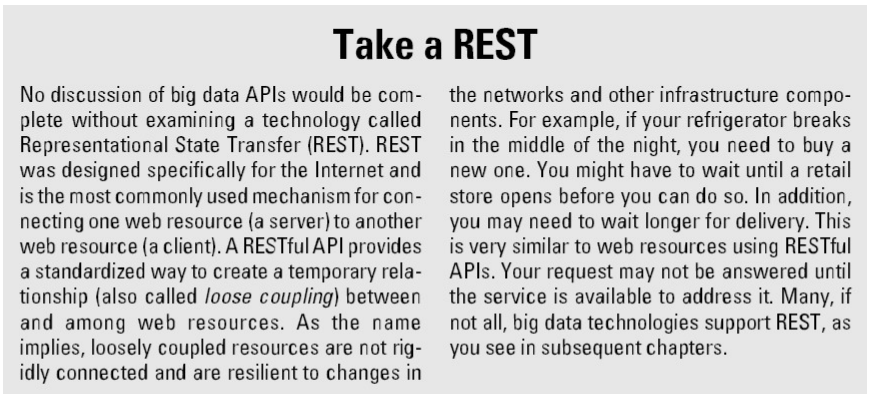
API (Application Programming Interface)

- API is the interfaces that provide bidirectional access to all the components of the stack — from corporate applications to data feeds from the Internet.
- Big data challenges require a slightly different approach to API development or adoption.
- A new technique, called Natural Language Processing (NLP), is emerging as the preferred method for interfacing between big data and your application programs.(Large Language Model (LLM) such as Chatgpt AI are now in popularity.)
- One way to deal with interfaces is to implement a "connector" factory

API: Connector Factory

- A “connector” factory adds a layer of abstraction and predictability to the process, and it leverages many of the lessons and techniques used in Service Oriented Architecture (SOA).
- Design a set of services to gather, cleanse, transform, normalize, and store big data items in the storage system. the factory could be driven with interface descriptions written in Extensible Markup Language (XML)
- This level of abstraction allows specific interfaces to be created easily and quickly without the need to build specific services for each data source.
- In practice, you could create a description of SAP or Oracle application interfaces using something like XML.
- Each interface would use the same underlying software to migrate data between the big data environment and the production application environment independent of the specifics of SAP or Oracle.
- If you need to gather data from social sites on the Internet (such as Facebook, Google+, and so on), the practice would be identical.

REST



Take a REST

No discussion of big data APIs would be complete without examining a technology called Representational State Transfer (REST). REST was designed specifically for the Internet and is the most commonly used mechanism for connecting one web resource (a server) to another web resource (a client). A RESTful API provides a standardized way to create a temporary relationship (also called *loose coupling*) between and among web resources. As the name implies, loosely coupled resources are not rigidly connected and are resilient to changes in the networks and other infrastructure components. For example, if your refrigerator breaks in the middle of the night, you need to buy a new one. You might have to wait until a retail store opens before you can do so. In addition, you may need to wait longer for delivery. This is very similar to web resources using RESTful APIs. Your request may not be answered until the service is available to address it. Many, if not all, big data technologies support REST, as you see in subsequent chapters.

Layer 2: Operational Databases

- Relational Database's ACID Characteristics

✓ **Atomicity:** A transaction is "all or nothing" when it is atomic. If any part of the transaction or the underlying system fails, the entire transaction fails.

✓ **Consistency:** Only transactions with valid data will be performed on the database. If the data is corrupt or improper, the transaction will not complete and the data will not be written to the database.

✓ **Isolation:** Multiple, simultaneous transactions will not interfere with each other. All valid transactions will execute until completed and in the order they were submitted for processing.

✓ **Durability:** After the data from the transaction is written to the database, it stays there "forever."

Not Only SQL Database

BASE
- Basically Available
- Soft State
- Eventual Consistency

- Distributed System
- CAP Theorem

Characteristics of SQL and NoSQL databases

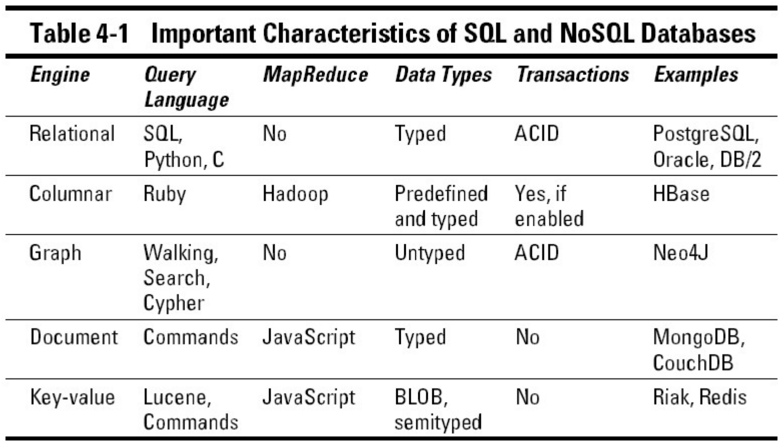


Table 4-1 Important Characteristics of SQL and NoSQL Databases

Engine	*Query Language*	*MapReduce*	*Data Types*	*Transactions*	*Examples*
Relational	SQL, Python, C	No	Typed	ACID	PostgreSQL, Oracle, DB/2
Columnar	Ruby	Hadoop	Predefined and typed	Yes, if enabled	HBase
Graph	Walking, Search, Cypher	No	Untyped	ACID	Neo4J
Document	Commands	JavaScript	Typed	No	MongoDB, CouchDB
Key-value	Lucene, Commands	JavaScript	BLOB, semityped	No	Riak, Redis

Layer 3: Organizing Data Services and Tools

- Many of these organizing data services are MapReduce engines, specifically designed to optimize the organization of big data streams.
- Organizing data services are, in reality, an ecosystem of tools and technologies that can be used to gather and assemble data in preparation for further processing.

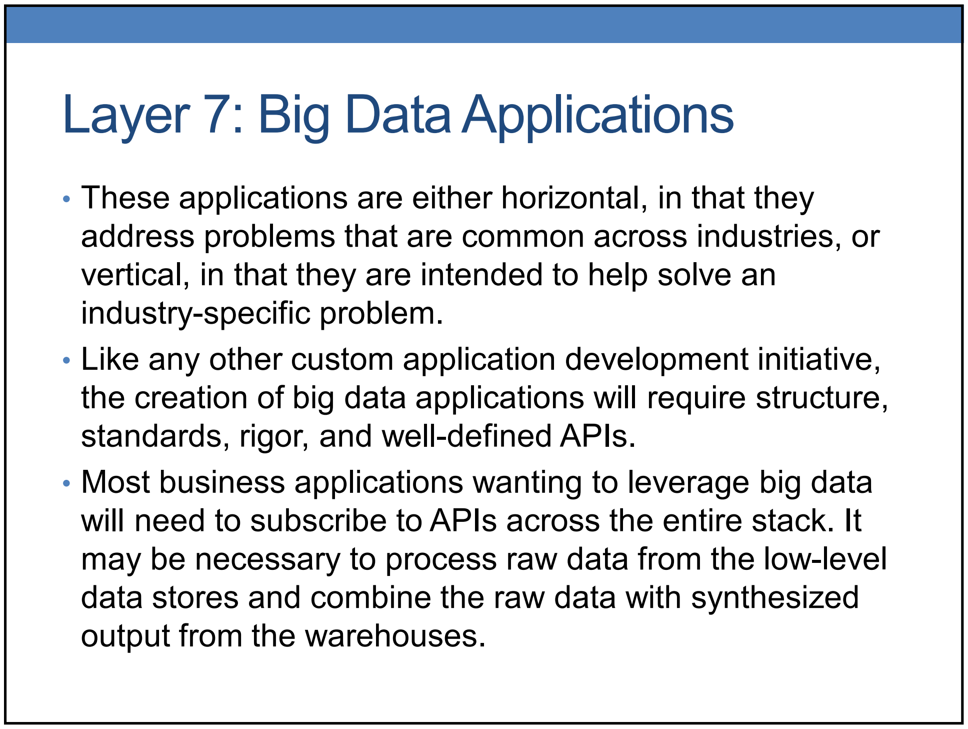
✓ **A distributed file system:** Necessary to accommodate the decomposition of data streams and to provide scale and storage capacity
✓ **Serialization services:** Necessary for persistent data storage and multilanguage remote procedure calls (RPCs)
✓ **Coordination services:** Necessary for building distributed applications (locking and so on)
✓ **Extract, transform, and load (ETL) tools:** Necessary for the loading and conversion of structured and unstructured data into Big Data File System such as Hadoop
✓ **Workflow services:** Necessary for scheduling jobs and providing a structure for synchronizing process elements across layers

Layer 4: Analytical Data Warehouses/ DataLake

- Data warehouses and marts simplify the creation of reports and the visualization of disparate data items. They are generally created from relational databases, multidimensional databases, flat files, and object databases — essentially any storage architecture. (Extract-Trasform-Load)
- Most data warehouse implementations are kept current via batch processing.
- The problem is that batch-loaded data warehouses and data marts may be insufficient for many big data applications.
- The stress imposed by high-velocity data streams will likely require a more real-time approach to big data warehouses.
- Data Lake is a new emerging concept for handling both Batch and Real-time Data which rely on ELT(Extract-Load-Transform)

Layer 5 & 6: Big Data Analytics and Visualization

- **Reporting and dashboards:** These tools provide a “user-friendly” representation of the information from various sources. These tools now access the new kinds of databases collectively called NoSQL (Not Only SQL).
- **Visualization:** Business users can watch the changes in the data utilizing a variety of different visualization techniques, including mind maps, heat maps, infographics, and connection diagrams. Sometimes, animation and interactive visualizations are used.
- **Analytics and advanced analytics:** These tools reach into the data warehouse and process the data for human consumption. Advanced analytics should explicate trends or events that are transformative, unique, or revolutionary to existing business practice. Predictive analytics and sentiment analytics are good examples of this science.



Layer 7: Big Data Applications

- These applications are either horizontal, in that they address problems that are common across industries, or vertical, in that they are intended to help solve an industry-specific problem.
- Like any other custom application development initiative, the creation of big data applications will require structure, standards, rigor, and well-defined APIs.
- Most business applications wanting to leverage big data will need to subscribe to APIs across the entire stack. It may be necessary to process raw data from the low-level data stores and combine the raw data with synthesized output from the warehouses.