DATA BUCKETS IN BIG DATA

A Thesis

presented to

the Faculty of Natural and Applied Sciences

at Notre Dame University-Louaize,Zouk Mosbeh

In Partial Fulfillment

of the Requirements for the Degree of

Master of Science in Computer Science

by

MÉLANIE MANSOUR

MAY 2019

Notre Dame University – Louaize, Zouk Mosbeh

Faculty of Natural and Applied Sciences

Department of Computer Science

We hereby approve the thesis of

Mélanie Mansour

Candidate for the degree of Master of Science in Computer Science

[illegible]	
Dr. Nazir Hawi	Supervisor, Chair
[illegible]	
Dr. Hoda Maalouf	Committee Member
[illegible]	
Dr. Maya Samaha	Committee Member

Declaration

The objective of my research is to classify the various forms of big data, distribute it, and remove the unnecessary ones. The term ‘big data’ usually refers to 3V’s, i.e., Volume (the amount of data), Variety (the types of data) and Velocity (the speed of data generated). My thesis research mainly emphases on addressing the complexity and scalability notions in big data.

The data mentioned in my paper was approved previously by my instructor Dr. Nazir Hawi. All the guidelines and ethics measures have been tailed properly while organizing my thesis.

This study was completed under the supervision of Dr. Nazir Hawi, Associate Professor of Computer Science, at Notre Dame University Louaize Zouk Mosbeh.

Signature:

[illegible]

Name: Mélanie Mansour

Date: May 20, 2019.

Abstract

Big data is an evolved term for large volume of unstructured, semi-structured and structured data having the potential to be used and mined for information in machine learning projects and other advanced analytics applications.

Big data is the new driver of the world societal changes and economic. The world’s data collection is reaching a tipping point for major technological changes that can bring different ways in finance, decision-making, cities, managing our health, and education. Latest technological improvements in computing, data handling, data storage, and trading have transformed the financial industry, hence growing liquidity, decreasing costs, and building new chances for business inquiries. While the data complexities are growing including data’s variety, value, velocity, volume, variability, veracity, the real impact hinges on our ability to discover the variety and scalability in the data through Big Data Analytics technologies. Due to the need of scalability as data technologies and volumes are increasing, fetching data is more time consuming, causing latency and encountering performance issues.

To manage and search data, we need efficient search methodologies. Proper indexing with multiple types and enormous data is not easy with the typical indexing used in databases. Hence, the proposed solution of buckets that chunks the data by type and criteria will make content-based multimedia retrieval systems more efficient and less time consuming.

Acknowledgments

First, I would like to express my sincere appreciation to my instructor, Dr. Nazir Hawi for the incessant support for my research study, for his motivation, immense knowledge and patience. His supervision guided me throughout my study. I would not ask for a better mentor and advisor for my research study. Besides Dr. Hawi, my deepest appreciation goes to Dr. Hoda Maalouf, Dr. Maya Samaha Rupert, Dr. Hicham Hage, Dr. Marie Khair and Dr. Pierre Akiki, for their insightful encouragement, comments, and for the inquiries they expressed that lead me to extend my study on several perspectives.

Table of Contents

List of Illustrations

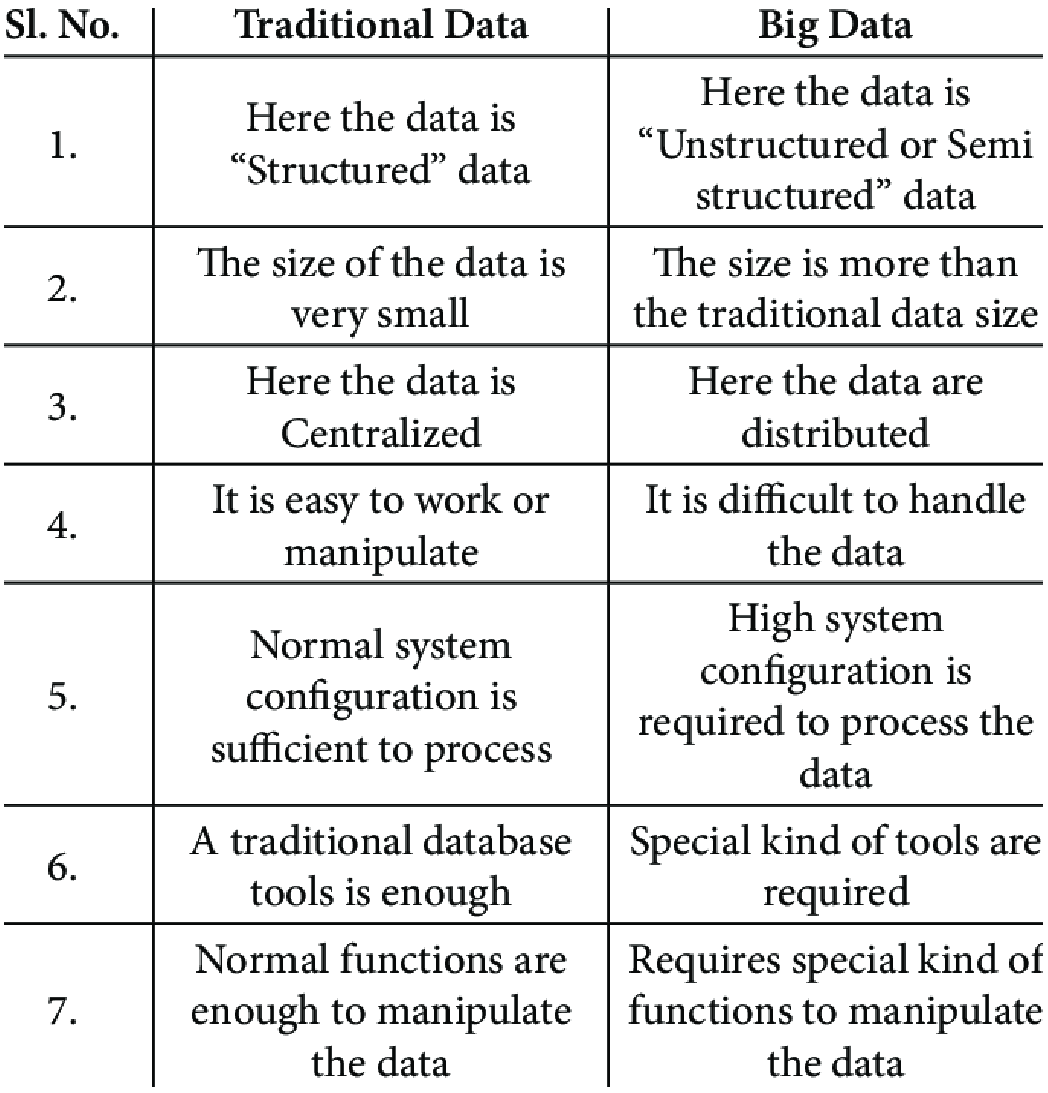
List of Acronyms

UI	User Interface
PHP	Hypertext Preprocessor
CRM	Customer Relationship Management
OS	Operating System
LBaaS	Load balancer as a service
I/O	Input/Output
ERP	Enterprise Resource Planning
ID	Identity
ROI	Return on Investment
HAProxy	High Availability Prox

Chapter 1: Introduction and Problem Definition

1.1 Introduction to the General Problem

When traditional handling techniques and data mining practices cannot discover the true meaning and insights of the data, then big data is used (Technopedia, (June4)). Large, time sensitive or unstructured data cannot be treated by relational database engines. That’s why big data is needed since it utilizes massive parallelism on the existing hardware.



Sl. No.	Traditional Data	Big Data
1.	Here the data is “Structured” data	Here the data is “Unstructured or Semi structured” data
2.	The size of the data is very small	The size is more than the traditional data size
3.	Here the data is Centralized	Here the data are distributed
4.	It is easy to work or manipulate	It is difficult to handle the data
5.	Normal system configuration is sufficient to process	High system configuration is required to process the data
6.	A traditional database tools is enough	Special kind of tools are required
7.	Normal functions are enough to manipulate the data	Requires special kind of functions to manipulate the data

Figure 1: Table Screenshot Traditional Vs. Big Data

Rajendran P., K. (2016) Reprinted from research gate from https://www.researchgate.net/figure/Comparison-of-big-data-vs-traditional-data_tbl1_307531952, copyright © 2016

Big data technology is definitely the solution for the fast growing need to store and manipulate current application data load. In fact, it aids in identifying more efficient ways of doing business, thus, costs can be reduced and the ROI on your investments can be increased.

Figure 2: Big Data
Retrieved from transform data into insight from https://tdwi.org/articles/2017/05/30/future-of-big-data-is-cloudy-and-bumpy.aspx, copyright © 2017

This is further optimized with quicker decision-making cycles since insights can be resultant in real-time, therefore growing the value of operations for every involved stakeholder. In fact, some major technology companies switched or are switching to big data technology and cloud computing. One example is Bank Audi Lebanon. Bank Audi is trying to switch to Big Data in order to identify potential risks associated with money lending processes in the bank, understand customer behavior based on the inputs received from their investment patterns, shopping trends, motivation to invest and personal or financial backgrounds, analyze and keep track of all the regulatory requirements by going through each individual application for accurate validation, assess the employee's performance whether or not have achieved the monthly/quarterly/yearly targets, aid in sifting through high volumes of data and respond to each of them adequately and swiftly.

The need for big data is based on six essential keys: (Anon, (2019))

1. Volume: Machine generated based on the amount of data.
2. Velocity: Large inflow of data, large inflow of relations and queries
3. Variety: Non-Traditional data, changes a lot based on user experience and requests
4. Value: this key is used to identify what is valid for transformation and extraction for data analysis.
5. Veracity: accuracy of data
6. Variability: variable data sources

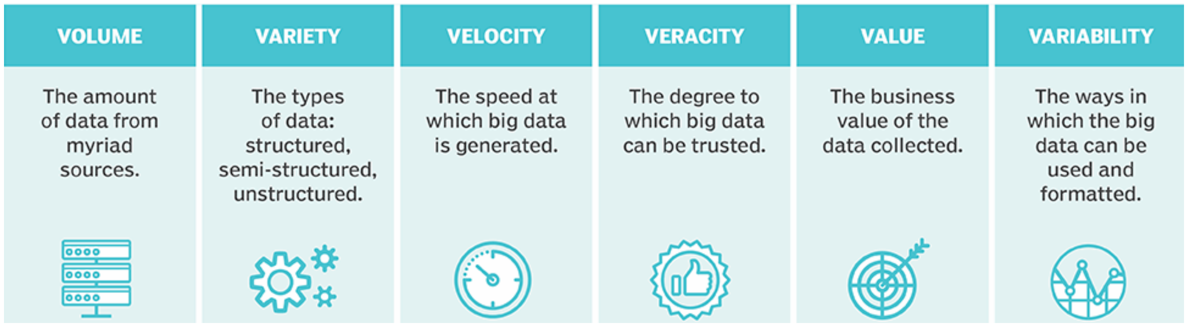

VOLUME
The amount of data from myriad sources.
VARIETY
The types of data: structured, semi-structured, unstructured.
VELOCITY
The speed at which big data is generated.
VERACITY
The degree to which big data can be trusted.
VALUE
The business value of the data collected.
VARIABILITY
The ways in which the big data can be used and formatted.

Figure 3: The 6 V's of Big Data.

However, with each new technology derives some complications, such as:

1. Heterogeneity of data: Data is classified as heterogeneous when it is composed of various formats and types.

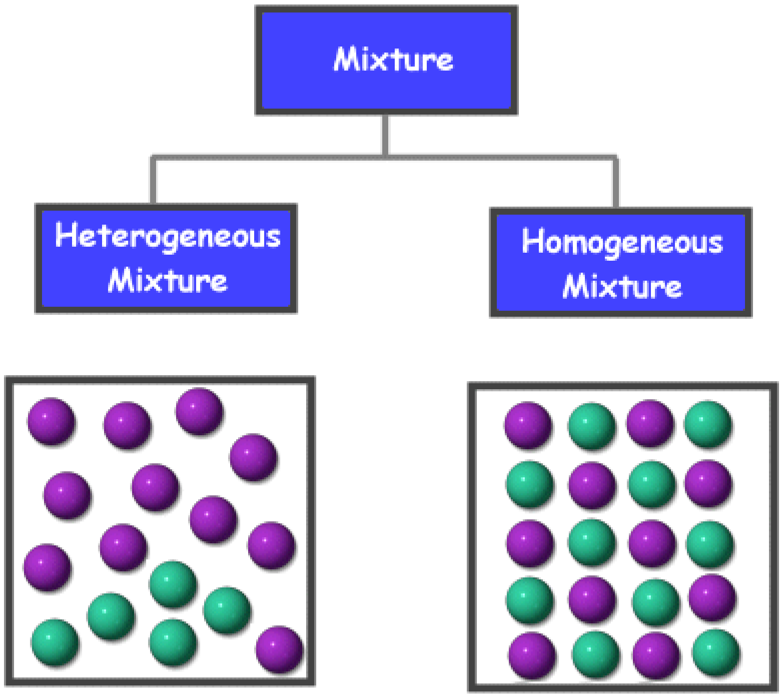


Figure 4: Heterogeneity vs Homogeneity
Retrieved from transform data into insight from https://byjus.com/chemistry/heterogeneous-mixture-homogeneous-mixture/, copyright © 2019

2. Scalability: is when a system handles growing amounts of workload by increasing resources to the system.

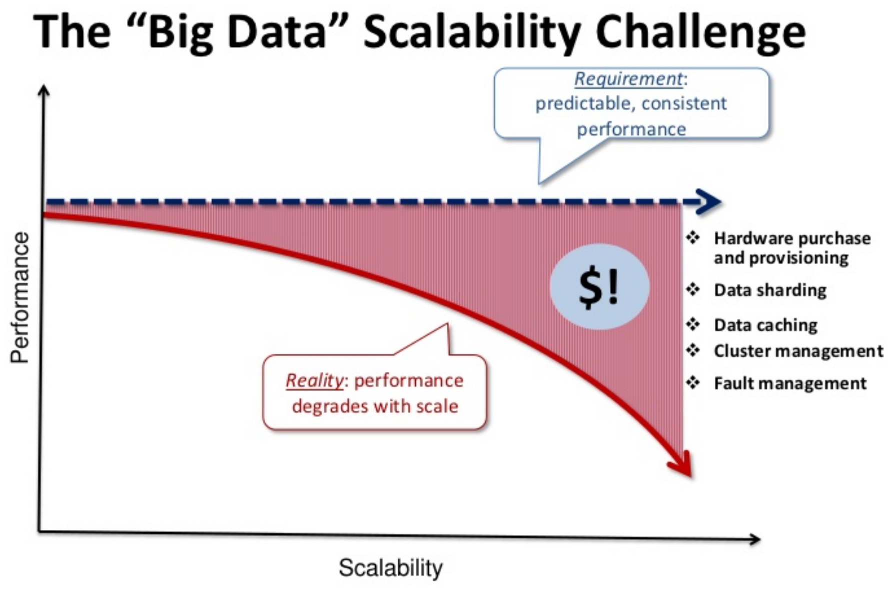


Figure 5: Figure Scalability
Retrieved from slide Share by Amazon Web Services from https://www.slideshare.net/AmazonWebServices/amazon-dynamodb/5-The_Big_Data_Scalability_Challenge, copyright © 2012

3. Timeliness: is about accessibility and availability of the data in decision-making. Well-organized, clean and clear data gives a stronger understanding of what to expect in the future.

Figure 6: Data Timeliness
Retrieved Reprinted from Measured Results Marketing by Christopher Antonopoulos from Panpote at FreeDigitalPhotos.net via https://www.measuredresultsmarketing.com/importance-data-quality-timeliness/, copyright © 2019

4. Complexity: is to accept the inexactness and complexity of data.

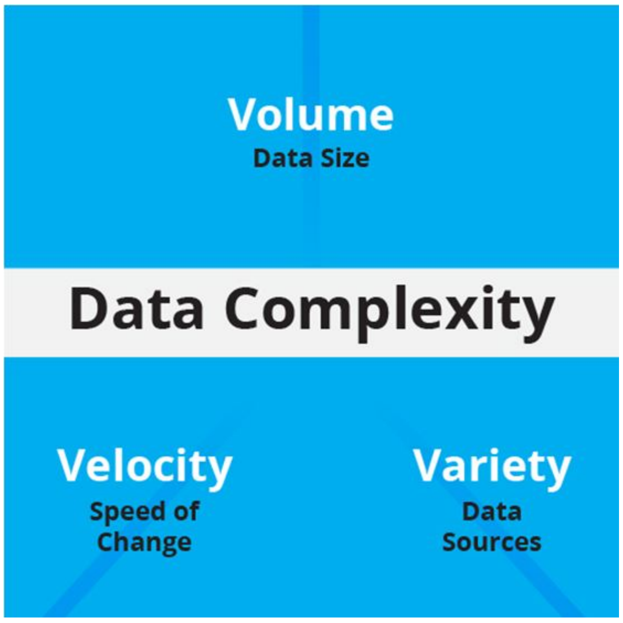


Figure 7: Complexity
Retrieved from Datamation by By Paul Rubens from: https://www.datamation.com/big-data/big-data-visualization.html, copyright © 2017

5. Privacy: Information privacy is the capacity of controlling the way personal information is utilized and collected. Privacy information is the ability of a group of individuals to keep their related data secret and private when performing data generation, processing, storage, access rights and other transactions over the internet.

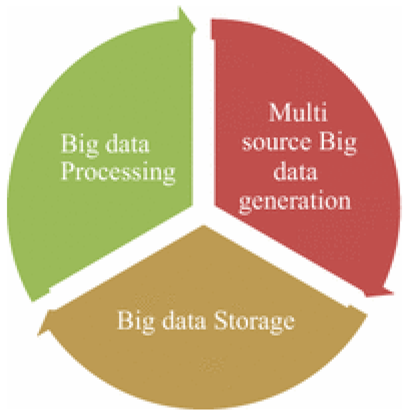


Figure 8: Big data life cycle stages of big data life cycle
Retrieved from Springer by Manasi Gyanchandani, Nilay Khare and Priyank Jain
From https://link.springer.com/article/10.1186/s40537-016-0059-y, copyright © 2016

In Figure 8 the Big data life cycle stages of big data life cycle is shown i.e., data generation, storage, and processing.

In this thesis, the notions of scalability and complexity (unstructured data) will be covered.

1.2 Problem Definition

The first obstacle faced in Big Data was its scalability. Two main approaches will be discussed in the scalability section which are: scaling up and scaling out where both will not answer the growing need of the immense client base. The second difficulty is with unstructured data. It is a major hindrance and an irritating problem for all database administrators. This issue is due to the lack of indexing in a proper method that will prevent getting the data swiftly and return the accurate and desired result from any search query. In fact, this part of the database querying is the most time consuming of all and presents the majority of latency problems.

1.3 Research Objectives

This thesis will cover two major objectives as listed below:

a) Prepare a scalability study where scaling up and scaling out work cohesively to extend the database and data files.

b) Create buckets to gather details on patterns for unstructured data to make the search inquiry faster and more accurate.

1.4 Approach and Main Results

The objective for this proposal is to balance the scalability approach where scaling up will be enhanced by improving the hardware and the backbone holding the distributed data files. Basically, the scalability approach will be made in harmony with the need of the application.

Regarding the structuring of the unstructured data, this study will elaborate and explain the buckets system where each paragraph of a text will be split into words and each word will be assigned to a bucket based on a very precise and clear criteria. In fact, this bucket will be used in a way that each word in the database will be indexed with a unique ID related to its respective bucket.

Chapter 2: Background and Motivation

Being an active member in the database administration field and having to deal with data constantly; people, employees and I find some problems in querying huge amounts of data especially if this data is unstructured, non-organized and centralized. Thus, a decision should be taken to contribute with a tailored solution for the frequent issues encountered in the field, hence making database administrator’s life easier and enhancing the applications performance.

2.1 Scalability

2.1.1 Scalability Definition

Managing, analyzing, storing and processing the overflow of data is done through a methodology called ‘scaling’. Data scaling is used in order to deal with enormous datasets. It is the ability to enlarge any type of data system to be able to handle large amounts of data, to enhance the performance and the response time of a system. (Isaacson, C. (2015)).

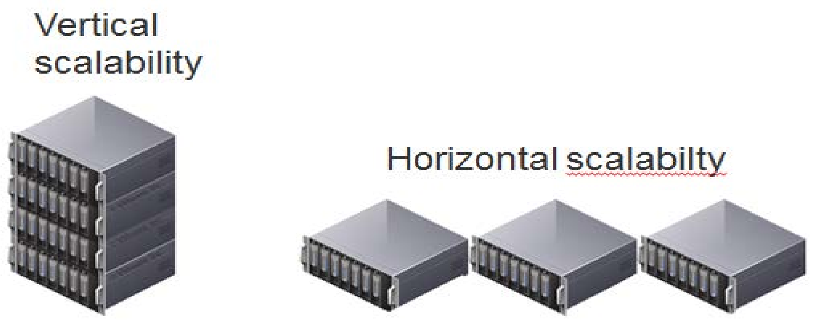
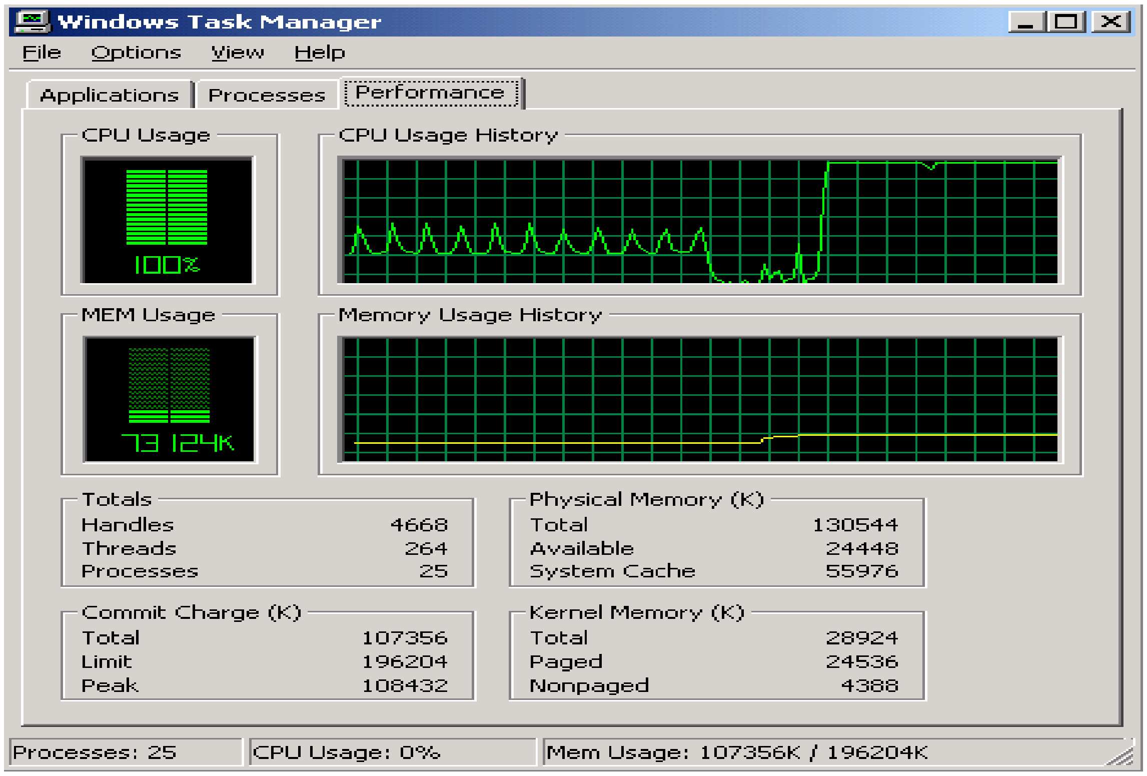


Figure 9: Scalability
Retrieved from Tech Target by Margaret Rouse from https://searchcio.techtarget.com/definition/vertical-scalability,copyright © 2019

Scalability platforms provide fast changes in the growth of volume, data or traffic. They use additional software, hardware to augment storage and output of the data. When performance issues rise, companies should consider implementing scalability in their enterprises. These problems can influence the customer retention, efficiency and workflow in a very bad way. High CPU usage, low memory and high disk usage are common bottlenecks of performance in scalability.

High CPU Usage

Erratic and slowness issues are performance indicators of high CPU usage that might be a harbinger to other problems. A user CPU is when a CPU is performing a productive work but requires server upgrades, whereas a system CPU is when the usage is consumed by the OS, and is commonly linked to I/O wait and the software, that is the idling time produced by the CPU waiting for the I/O subsystem.



Totals

Handles	4668
Threads	264
Processes	25

Physical Memory (K)

Total	130544
Available	24448
System Cache	55976

Commit Charge (K)

Total	107356
Limit	196204
Peak	108432

Kernel Memory (K)

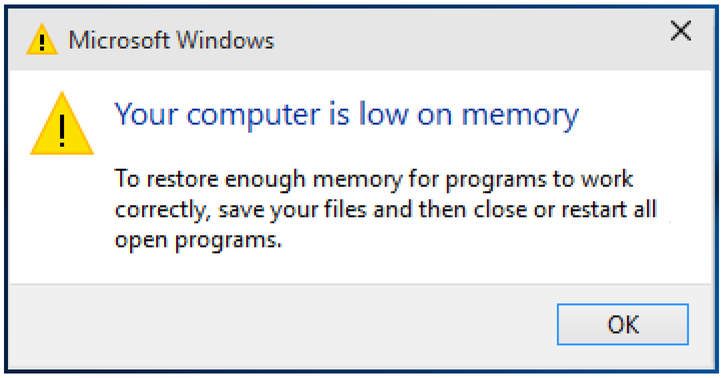
Total	28924
Paged	24536
Nonpaged	4388

Processes: 25 CPU Usage: 0% Mem Usage: 107356K / 196204K

Figure 10: High CPU Usage
Retrieved from http://windows-exe-errors.com/how-to-fix-cisvc-exe-high-cpu-usage/, copyright © 2019

Low Memory

Lack of memory in the servers to handle the data load of the application can completely slow down the application. A RAM upgrade can be a solution for the low memory, however it might produce a memory leak; then repairing and finding the leak in the code of the application is a must.


Microsoft Windows

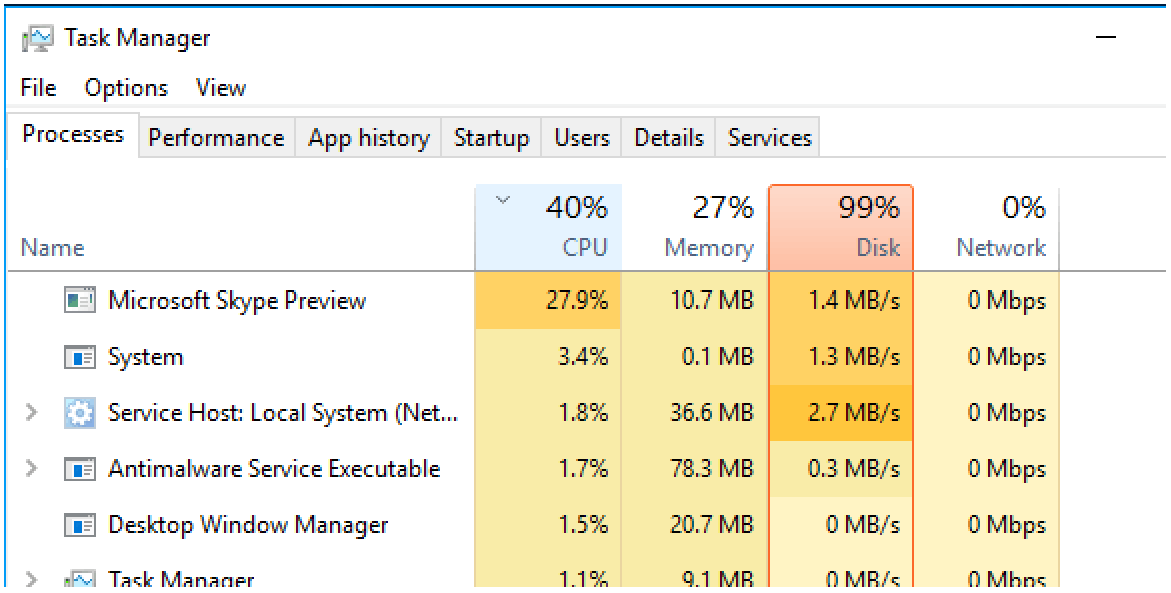
Your computer is low on memory

To restore enough memory for programs to work correctly, save your files and then close or restart all open programs.

Figure 11: Low Memory
Retrieved from Driver Easy by April Cai from https://www.drivereasy.com/knowledge/windows-10-computer-low-memory, copyright © 2018.

High Disk Usage

It is usually caused by maxed out disks, a point at which no more, profit, improvement or benefits occurring; which is why the data scaling is needed.


Task Manager
File Options View
Processes Performance App history Startup Users Details Services

Name	40% CPU	27% Memory	99% Disk	0% Network
Microsoft Skype Preview	27.9%	10.7 MB	1.4 MB/s	0 Mbps
System	3.4%	0.1 MB	1.3 MB/s	0 Mbps
Service Host: Local System (Net...	1.8%	36.6 MB	2.7 MB/s	0 Mbps
Antimalware Service Executable	1.7%	78.3 MB	0.3 MB/s	0 Mbps
Desktop Window Manager	1.5%	20.7 MB	0 MB/s	0 Mbps
Task Manager	1.1%	9.1 MB	0 MB/s	0 Mbps

Figure 12: High Disk Usage
Retrieved from Onine Tech Tips by Aseem Kishore from Onine Tech Tips from https://www.online-tech-tips.com/windows-10/troubleshoot-100-disk-usage-windows-10/, copyright © 2016

There are two kinds of scaling: **scale up** and **scale out**.

The focus in Big Data is in the size of data handling. Managing large and rapidly increasing volumes of data has been and still is a challenging issue for many decades. In the past, this delema was resolved by upgrading the processing power to provide the resources needed to adapt with the massive increase of data volume.

However, the data volume is scaling much faster than computer resources are. This is why merging between scaling up and scaling out is a must in order to handle the increasing data volume.

Scale up

It means enlarging the available processing power and fastening the processing time needed to analyze the data existing in the big data platform.

In the past five years, the processor technology has made a dramatic change when the clock cycle frequency was doubled. As a result, the clock speed was tremendously delayed and the processors were built with an increasing number of cores.

Previously, parallelism across nodes was a big problem for a processor's creation, now parallelism within a single node is the major issue. Moreover, the transfer to packing multiple sockets adds an additional complexity for the intra-node parallelism. In addition, the "dark silicon" (significant amount of chip resources) will prohibit users from using all of the hardware in the system continuously and the data processing systems will likely have to actively manage the power consumption of the processor.

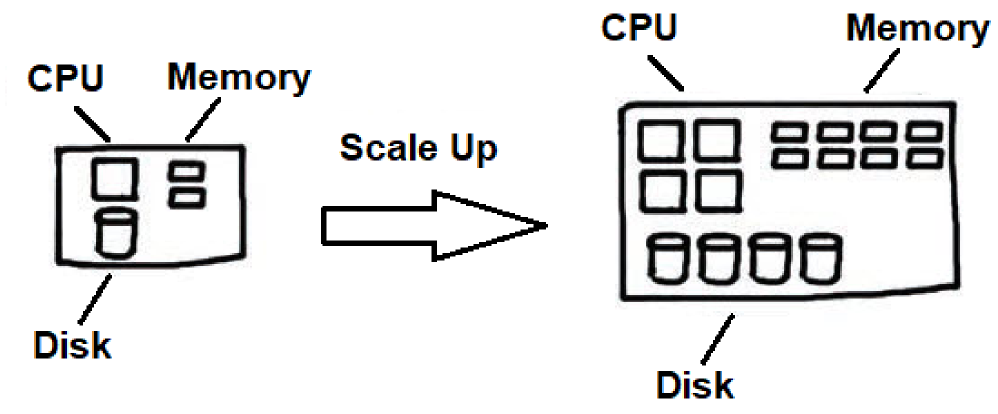


Figure 13: Illustration of the scale-up process

Scale out

It refers to horizontal scaling, which means adding additional servers for the application, and operate a distributed environment capable of parallel computing. In fact, scaling out is switching from a single server to many servers. If it is performed in an appropriate manner, then it can be considered as a long-term solution for the application's performance. However, going from a simple monolithic computing environment to a scalable cluster is a huge step.

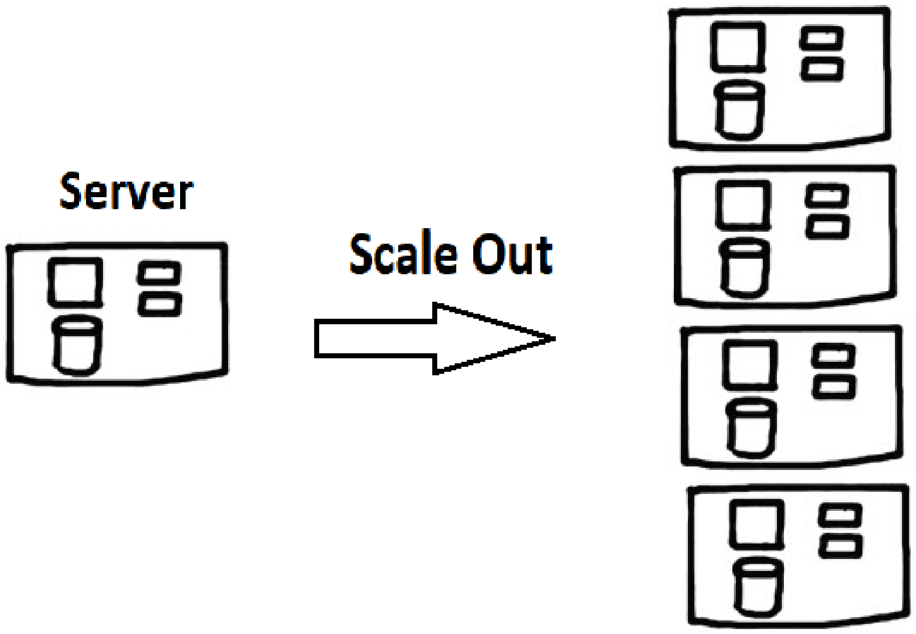


Figure 14: Illustration of the scale-out process

2.1.2 System Architecture

2.1.2.1 Data Acquisition

The problem in big data begins in data acquisition and in the decision of wether to keep or discard the data, specially that they are stored in a none native structured format. For example: tweets, comments or blogs are limited in text structure while images and videos are structurely tailored. Furthermore, images and videos optimize the storage and display, but are not built for semantic context and search, that is nowadays a big concern for the norrmal internet surfer. Thus, a balance between structured data and unstructured data should be found to optimize both storage and search in the new system.

What is
Data Acquisition
It is the process of gathering, filtering and cleaning the data before it is put in a data warehouse or any other storage solution on which data analysis can be carried out.
Data acquisition is one of the major big data challenges

Figure 15: Data Acquisition
Retrieved from Slide share by Prompt Cloud from : https://www.slideshare.net/promptcloud/5-most-common-mistakes-of-data-acquisition, copyright © 2016

2.1.2.2 Load Balancer Tier

The load balancer distributes efficiently the incoming traffic of the network on a group of servers known as server pool or server farm.

Websites can serve huge amounts of concurrent requests from clients or users returning correct texts, videos, application data or images in a reliable and a rapid manner.

The load balancer is often called a traffic cop since it sits in front of the servers and routes client requests across all servers. In fact, it is capable of accomplishing those requests in a way that increases capacity and speed utilization, making sure that no server is overworked since it can cause performance degradation. If a server is down, then the load balancer will redirect the traffic to the remaining available servers.

In this manner, the load balancer achieves the below jobs:

- Allocates network load or client requests in an efficient manner across several servers
- Ensures high reliability and availability since it only sends requests to online servers
- Offers the flexibility to subtract or add servers as demanded

Incoming traffic from end-user Web browsers normally gets to a load balancer as the first tier. The load balancer’s job is to equilibrise the load for requests across the next tier and the application server tier as well.

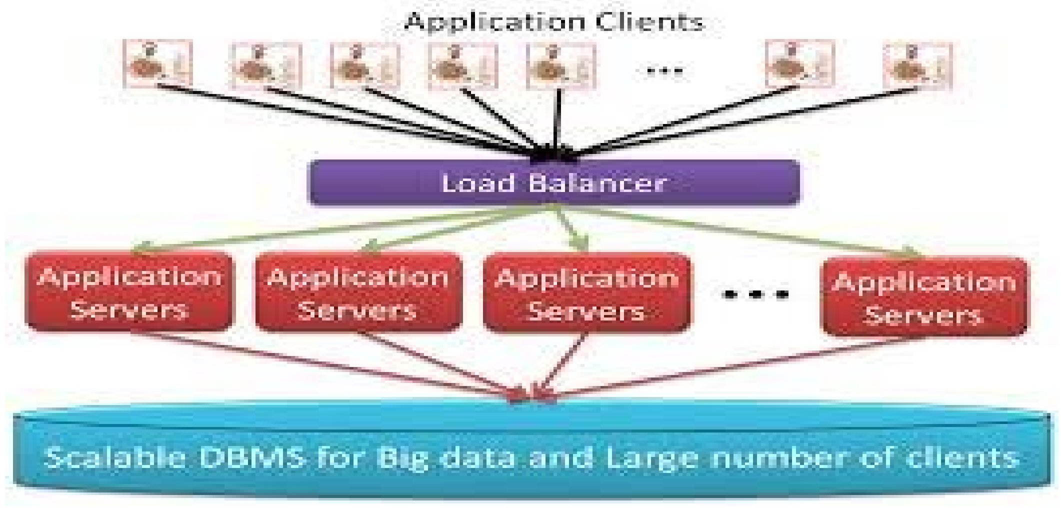


Figure 16: Load Balancer
Retrieved from University of California, Santa Barbara by Amr El Abbadi, Divyakant Agraw, Sudipto Da from: https://openproceedings.org/2011/conf/edbt/AgrawalDA11.pdf, copyright © 2011

There are different types of load balancers, some are **software-based** while others are **hardware based.**

Software-based load balancers are categorized into two general types: Load Balancer as a Service (LBaaS) and installable load balancers. (Taral Shah,(2019))

Stratoscale Symphony LBaaS and AWS ELB are examples of LBaaSs.

They are managed by cloud providers and offer inherent elasticity, fault tolerance, management and installation.

Nginx, HAProxy , LVS and Varnish are examples of installable software load balancers. These load balancers need installation, management as well as configuration. The user has to plan how to scale the load balancer and deal with fault tolerance.

Software-based load balancers are further classified according to the routing algorithms as Round-robin Scheduler, weighted scheduler, and Least Connections First balancers scheduler.(StratoScale,(2019))

- **Round-robin load balancers** are beneficial when servers have the same amount of memory and compute. Requests are sequentially sent to each server one by one. AWS Application Load Balancer and AWS Classic Load Balancer are examples of round-robin load balancers.
- **Weighted load balancers** are utilized when there exist resources of various types such as, three servers with various amount of memory and compute. In that case, directing the traffic towards the servers having higher amount of memory compute will be ideal. The other servers will get fewer traffic. Fortinet Weighted Load Balancers is an example of weighted load balancers.

- **Least Connections First Balancers** follows the notion of the least connections first algorithm. In fact, the request will be sent to the servers having the least number of connections. This type of load balancers is utilized when we want to manage sticky sessions. Citrix NetScaler is an example of least connection first balancers.

Hardware based load balancer is a type of a hardware device with a particular OS distributing the traffic of a web application across cluster of application servers in a way that application servers will not be overworked**.** (Avi Networks, (2011))

Usually, application servers and hardware based load balancers are installed in on-premises data centers and the number of load balancers depends on the estimated amount of peak traffic. Load balancers are generally deployed in pairs in case one failed. If the application runs on a Cloud , the user can benefit from the provider's load balancer existing in his infrastructure. Irrespective of the type of the load balancer you use, the functionality is basically the same.

Load balancers do not store any state or data, thus they are said to be **stateless**. Altough, there are some exceptions to this, depending on the routing method supported by the load balancer. Even in these cases, the information is minimal and can be easily rebuilt when required.Most load balancers provide different rules for distributing the data load. The two most common rules are round robin routing and sticky routing.

Round robin routing (Avi Networks, (2011)) distributes user requests one at a time to an application server for processing, allowing requests to be serviced in equal amount of time. It is a simple way to distribute client requests across a group of servers. A client request is forwarded to each server in turn. The algorithm instructs the load balancer to go back to the top of the list and repeats again. Round Robin is easy to implement and conceptualize; round robin is the most widely deployed load-balancing algorithm. Using this method, client requests are routed to available servers on a cyclical basis. Round robin server load balancing works best when servers have roughly identical computing capabilities and storage capacity.

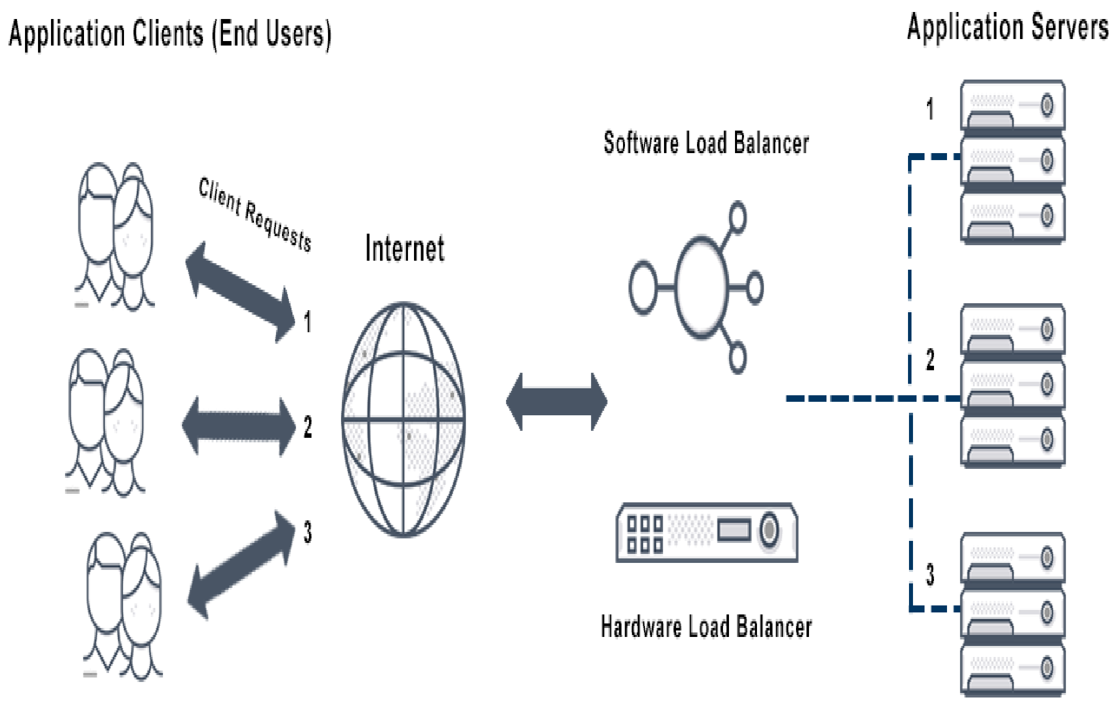


Figure 17: Round Robin Server Load Balancing
Retrieved from Avi Networks from https://avinetworks.com/glossary/round-robin-load-balancing/, copyright © 2011

The disadvantage of this approach is that each request, even if it is from the same user session, will potentially go to a different application server for processing. Sticky routing processes all transactions or requests for a user session by the same application and routes other requests to the original application server for processing, allowing the application to retain session state, the information about the user and their actions.

Sticky routing (Avi Networks, (2011)) is very useful in some scenarios, and the session state can ultimately allow a better experience for the end user.A load balancer that keeps sticky sessions will create a unique session object for each client. For each request from the same client, the load balancer processes the request to the same web server each time, where data is stored and updated as long as the session exists. Sticky sessions can be more efficient because unique session-related data does not need to be migrated from server to server. However, sticky sessions can become inefficient if one server accumulates multiple sessions with heavy workloads, disrupting the balance among servers. If sticky load balancers are used to load balance round robin style, a user's first request is routed to a web server using the round robin algorithm. Subsequent requests are then forwarded to the same server until the sticky session expires, when the round robin algorithm is used again to set a new sticky session. Conversely, if the load balancer is non-sticky, the round robin algorithm is used for each request, regardless of whether or not requests come from the same client.

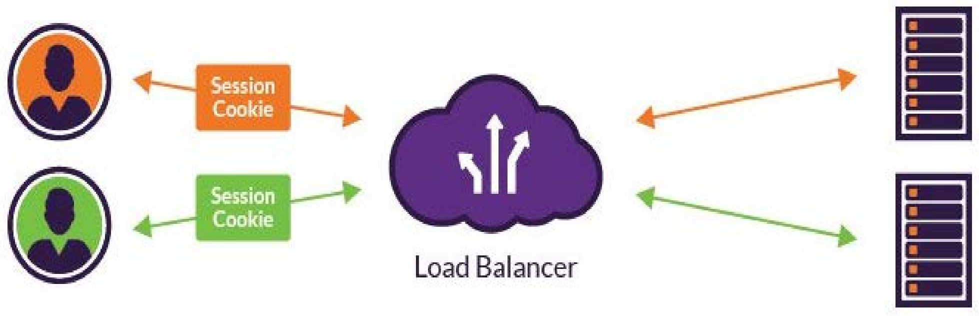


Figure 18: Sticky Routing Server Load Balancing Avi Networks
Retrieved from Avi Networks from https://www.imperva.com/learn/availability/sticky-session-persistence-and-cookies/, copyright © 2011

2.1.2.3 Application Server Tier

The application server includes everything from the UI (user interface) itself to all the functions and code snippets to processing the user requests. In many applications, it is desirable to separate the user interface from the application logic, enabling an additional application service tier for processing application transactions or requests. This is the ideal method of structuring the application server tier, but in some languages like PHP (Hypertext Preprocessor), the UI and service tier can be combined if desired.

Today's application servers can run applications written in almost any programming language, including PHP, Java, Ruby, Python etc... Furthermore, there are numerous application frameworks built for these languages to make the job easier and faster. For the purpose of discussing Big Data scalability, the specific language or framework is mainly insignificant. Generally, the application server is not required to store state, and can be considered stateless (XenonStack, (2018)).

The notable exception is the session state described above. This should be limited to a relatively small amount of data, and if a user's session is lost, it can be easily restored or rebuilt when a user creates a new session.

2.1.2.4 Database Tier

Let's now shift to the core of the application architecture, the database tier, where all the states should be saved and logged. In fact, the application data stored in a database must be a permanent record of actions performed by end users, otherwise the application servers, or the application itself cannot function.

In fact, the database tier can store gigabytes to terabytes and for some organizations even petabytes. Therefore, a database can contain many states related to various users' session. HTTP is a stateless protocol that allows applications to distribute resources across more than one web server. This allows an application to distribute requests across many web servers, thus dividing the load and permitting scaling of the application.

For example, an application like Amazon wants to hold the contents of a shopping cart using session variables. An HTTP request submits an order and is processed by reading the session variables that holds the state of the cart. Moving the session data to the database lets the application to scale horizontally at the middle tier. The web server does not need to store session variables, so the HTTP requests can be processed by various web servers.

Thus, the necessity for Big Data scalability, to accommodate large data sets supporting a variety of high volume application is needed.

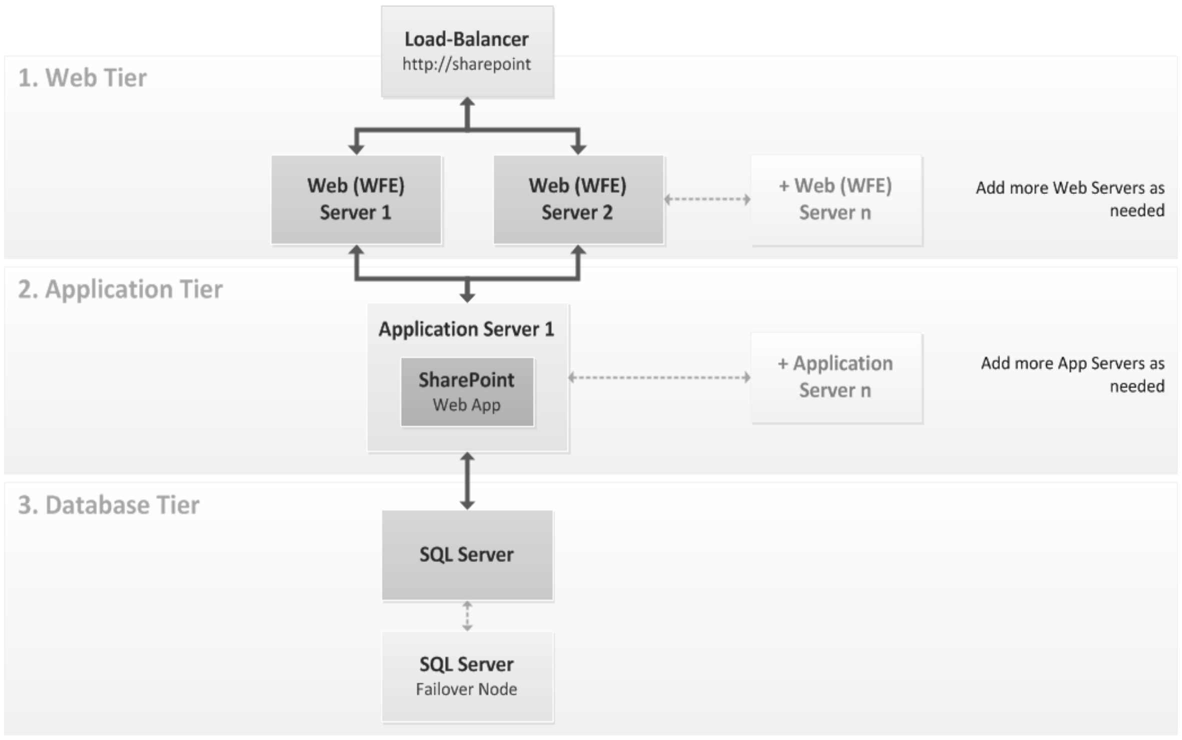


Figure 19: 3-tier Architecture
Retrieved from Paula It's Blog from http://paulasitblog.blogspot.com/2014/08/deploying-sharepoint-2013-3-tier.html, copyright © 2016

2.2 Unstructured Data

The propagation of unstructured data keeps on growing within all types of organizations.

So how to manage effectively this increasing flow of information?

By using an effective solution through indexing and searching techniques.

Mentioning unstructured data is referring to emails, images, documents, video, and audio and so on. In this framework, content-based and text-based approaches are incorporated for retrieving unstructured data. This retrieval structure can support different types of queries and might accept metadata-based documents and multimedia examples.

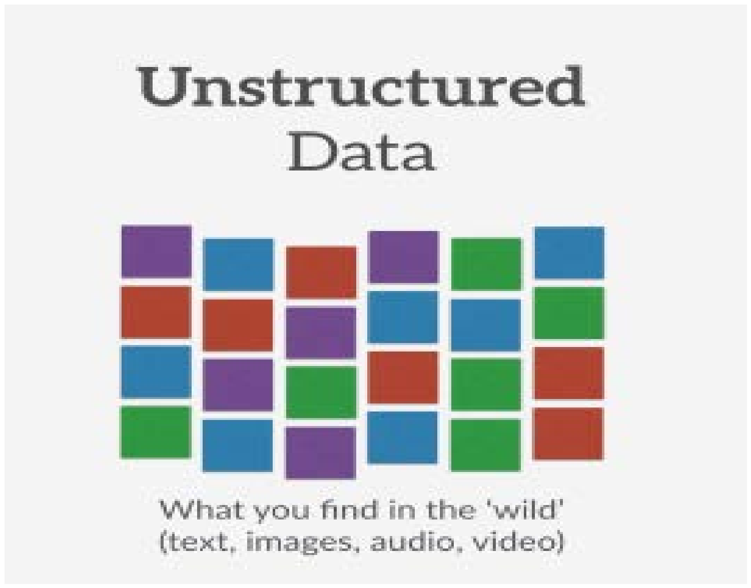
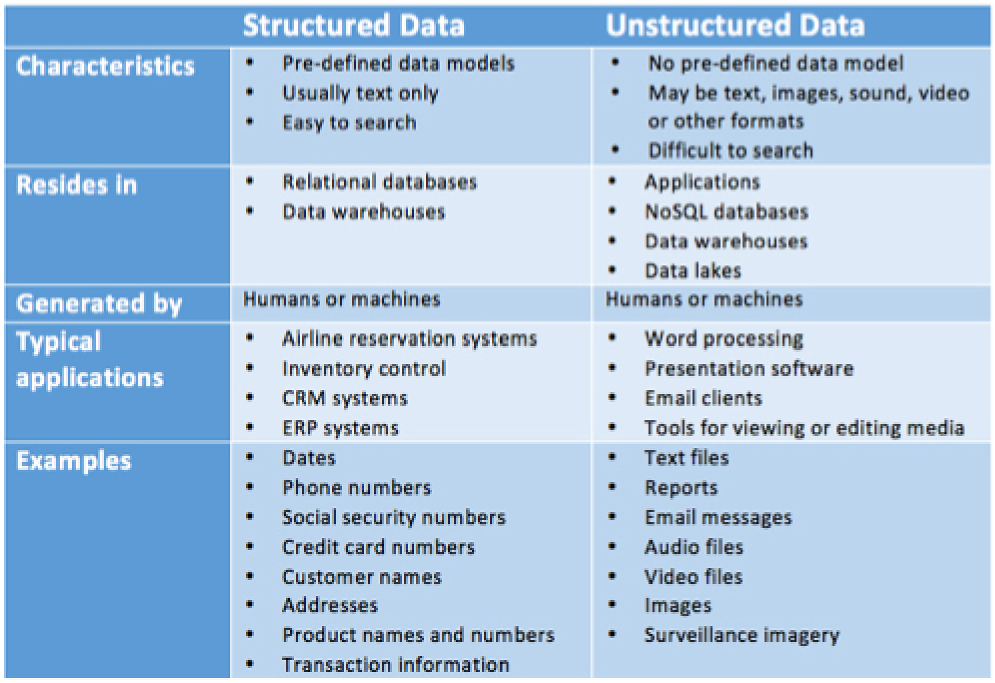


Figure 20: Structured vs. Unstructured data
Retrieved from Big Data from http://bigdata.black/infrastructure/storage/unstructured-data/ , copyright © 2016

2.2.1 Overview

Obviously, the internet is an enormous unstructured data collection, which makes it very hard to retrieve and search valuable information. Due to the huge number of unstructured data, search engines that rank and search documents containing unstructured data based on their significance to user queries become vital for information seeking.

Search engines are essential in determining significant documents within a short period of time, rendering a high search efficiency as one of the key design and implementation objective of a search engines. Therefore, efficient indexing methods that organize documents according to their contents is needed.



	Structured Data	Unstructured Data
Characteristics	• Pre-defined data models • Usually text only • Easy to search	• No pre-defined data model • May be text, images, sound, video or other formats • Difficult to search
Resides in	• Relational databases • Data warehouses	• Applications • NoSQL databases • Data warehouses • Data lakes
Generated by	Humans or machines	Humans or machines
Typical applications	• Airline reservation systems • Inventory control • CRM systems • ERP systems	• Word processing • Presentation software • Email clients • Tools for viewing or editing media
Examples	• Dates • Phone numbers • Social security numbers • Credit card numbers • Customer names • Addresses • Product names and numbers • Transaction information	• Text files • Reports • Email messages • Audio files • Video files • Images • Surveillance imagery

Figure 21: Structured versus Unstructured data Characteristics
Retrieved from Datamation by Christine Taylor from https://www.datamation.com/big-data/structured-vs-unstructured-data.html, copyright © 2018

2.2.2 Textual or non-textual data

1) E-mail messages, PowerPoint presentation, word documents, instant messages and collaboration software are textual unstructured.
2) MP3 audio files, JPEG images, and flash video files are classified as Non-textual unstructured data.

Unstructured information is normally text-heavy but can contain data such as numbers, dates, multimedia data and facts as well.

2.2.3 Proposed System Architecture

The proposed system includes content-based approaches that hold a full-text search for textual data, audio, video and image data.

Multimedia search engines can look for textual data, audio, video texts that is classified as unstructured data. These engines utilize content-based multimedia retrieval approach to combine content-based image retrieval, content-based audio retrieval, content-based text retrieval, and content-based video retrieval approaches for effective searching and indexing.

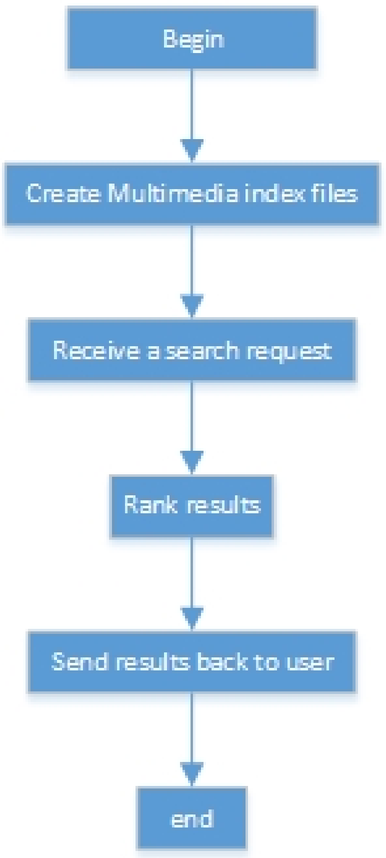


Figure 22: Functional model of multimedia search engine.

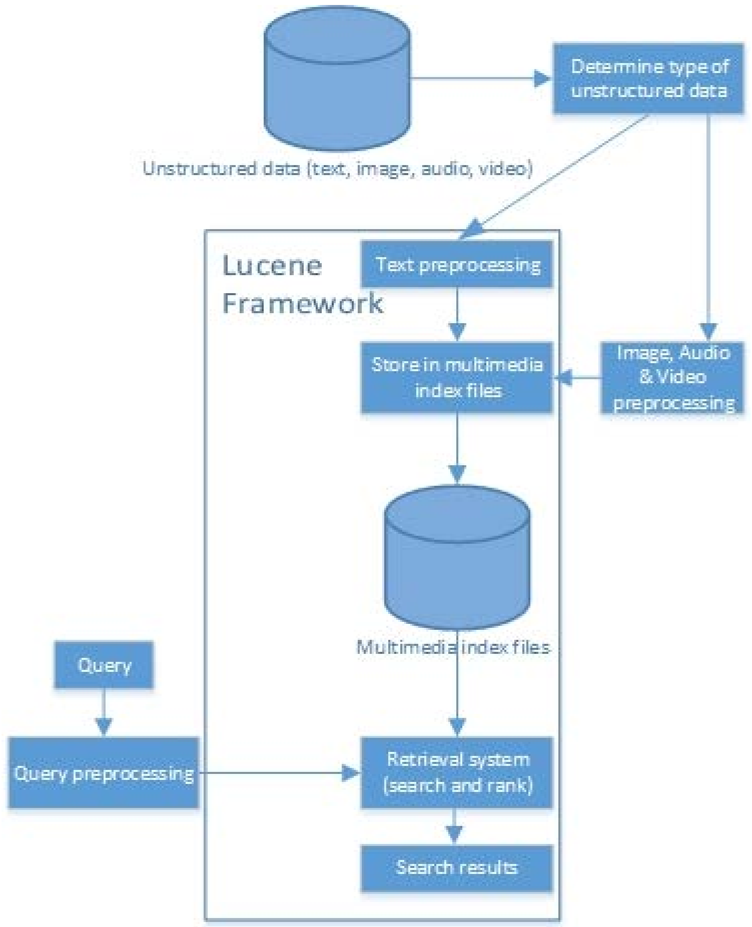

Unstructured data (text, image, audio, video)
Determine type of unstructured data
Lucene Framework
Text preprocessing
Store in multimedia index files
Image, Audio & Video preprocessing
Multimedia index files
Query
Query preprocessing
Retrieval system (search and rank)
Search results

Figure 23: Unstructured Data Indexing and retrieval framework

First, the system detects the type of unstructured data.

Second, if the data is of type text, then the text preprocessing operations are executed by a text-preprocessing module.

Text preprocessing

Lucene (Sonawane,A.(2009)), a high performance, full featured text search and open source engine library written in java implemented the preprocessing module. Lucene is a suitable technology that adds full-text search functions to any application. It can be used to build search capabilities for applications such as mailing lists, database search, e-mail clients, Web searches etc.…Web sites like The Server Side ,Wikipedia,jGuru, and LinkedIn have been powered by Lucene. Lucene has many features. It:

a) Has accurate, powerful and efficient searching algorithms.
b) Calculates a specific score for each document that matches a given query and returns the most relevant documents ranked by the scores.
c) Supports various powerful query types, for example WildcardQuery, RangeQuery, FuzzyQuery, PhraseQuery, BooleanQuery, and many more.
d) Aids in analyzing human-entered query expressions.
e) Able users to spread the searching behavior using filtering, custom sorting and query expression parsing.
f) Utilizes a file-based locking mechanism to avoid simultaneous index alterations.
g) Permits indexing and searching concurrently.

Image preprocessing

1) Determine file type (for example: TIFF file types ends in .tif , JPEG file types ending in .jpg etc.)

2) Convert to standard image type such as GIF.

3) Use standardized image size.

4) Feature extraction is the most essential step in image classification. It is used to attain efficient retrieval, aids in extracting the feature of an image in an ideal manner. Feature extraction techniques are applied to get the feature that will be useful in recognizing and classifying the images.

Audio preprocessing

Audio preprocessing converts audio data to text data in order to be accessible via Lucene indexing mechanisms in an easy way.

1) Determine audio data type (for example: AU/SND Files,WAVE Files,AIFF Files,MP3 Files)

2) Convert to standard format (frequency normalization)

3) Speech recognition (to detect words in the audio data)

Video preprocessing

1) Determine video data type (for example: .mov, .wmv, .ogg, .avi, .flv, .ogg, .mp4, .mpeg)

2) Convert video to standard type such as MP4

3) Process and extract audio data by audio preprocessing methods

4) Process and extract image-by-image preprocessing methods.

Query preprocessing

It includes both text preprocessing and image preprocessing.

The user query can be keyword query, metadata query, simple text query and exemplar image query.

Lucene mechanisms can support several query types for example field search, Boolean search, wildcard search, range search and fuzzy search.

Users might look for images or videos by text or search for audio data via text.

2.3 Research Motivation

This world is driven by data communication and transfer. In a broad range of application areas, data is being collected at unprecedented scale. Decisions previously based on guess or reality example are now based on the data collected itself. While the potential benefits of Big Data are real and significant, some initial success stories have already showed the power of Big Data, however there are other challenges for the utter success of this new technology mentioned earlier in the introduction which are: Variety, Velocity, Value and Volume.

The analysis of Big Data involves multiple distinct phases as listed below:

1. Acquisition/ Recording is the actual gathering and collection of data sets across various channels and sources.
2. Extraction/Cleaning/Annotation since raw data collected is usually not in a format ready for analysis, it wishes to pull out the required information from its original sources, and in turn express this in a structured form.

3. Integration/Aggregation/ Representation combine various types of data (e.g. sensor, video weather data) formatting the data into a standard form to make easier querying and representation.
4. Analysis/ Modeling to process collected data plus have a full vision of it. For example, recognizing value trends of various measures within data or classifying data based on its values.
5. Interpretation is to clarify and understand big data.

In all these phases, many factors are to be taken into consideration such as:

a. **Heterogeneity** is easily accepted when humans consume information. Actually, the richness and subtly of natural language can offer valued depth. However, machine analysis algorithms, (expect homogeneous data) cannot understand this constraint. Hence, data must be wisely structured as an initial step in data analysis. Some errors and incompleteness in data are probable to persist even after error correction and data cleansing. These errors and incompleteness should be managed throughout data analysis.
b. **Scale size** is the initial thing anyone thinks of in Big Data. A challenging issue for several decades was handling rapidly and large increasing volumes of data. Previously, this issue was mitigated by processors getting faster, following Moore's law, to offer the needed resources to handle growing volumes of data. A major shift underway now is that data volume is scaling faster than computer resources, and the CPU speeds are static.

c. **Timeliness** is when the time taken for analyzing the data is relative to the size of the data set to be managed. The design of the system that deals with the size of the data will also result in a system that can manage a given size of data set faster.

d. **Privacy** is also a huge concern, and rises only in the context of Big Data. The inappropriate usage of personal data, due to linking data from multiple sources is a serious issue. Handling privacy is problematic and needs to bse addressed from both sociological and technical perspectives to realize the promise of big data. An important way is to reconsider security for information sharing in Big Data use cases.

e. **Human Collaboration** can be simply detected; however, computer algorithms find a rough time discovering despite the great advances made in computational analysis. Idyllically, analytics for Big Data won't be all computational; rather it shall be explicitly designed to have a human in the loop.

Chapter 3: Original Work

3.1 Introduction

All the previously generated methods and algorithms (scalability, structuring unstructured data) tackle one aspect of the problem and ignore the other. For example, image indexing facilitates the search for an image based on many factors and parameters that makes the result returns in a fraction of a second but it doesn't resolve the fact that the search engine search the whole available data to match the query parameters and search for similarities in it to return an accurate result for the user. Other methods did resolve the data storage issues but didn't take into consideration the data indexing where they split the data into even entities without any filters such as the scale out approach where many servers contains parts of the database without any sort of grouping.

3.2 Data Buckets

Bin sort or bucket sort (En.wikipedia.org.(2019)), is a sorting algorithm(an algorithm that puts elements of a list in a specific order) that allocates the elements of an array into a number of data buckets(a type of document or data buffer in which data is divided into different parts).

Figure 24: Bucket of data
Retrieved from Transform Data Into Insight by Barry Devlin, from https://tdwi.org/articles/2016/02/01/data-warehousing-30.aspx, copyright © 2016

Each bucket is then sorted separately, either by applying a different sorting algorithm, or by applying recursively the bucket sorting algorithm. Bucket sort is a distribution sort, a generalization of pigeonhole sort(a sorting algorithm suitable for sorting lists of elements where the number of elements (n) and the length of the range of possible key values (N) are approximately the same), and is a cousin of radix sort(a non-comparative integer sorting algorithm that sorts data with integer keys by grouping keys by the individual digits which share the same significant position and value) in the most-to-least significant digit flavor.

Bucket sort can be applied with comparisons; thus can be considered a comparison sort algorithm (reads only the list elements through a single abstract comparison operation often a "less than or equal to" operator such as A<B) that determines which of two elements should occur first in the final sorted list.). The computational complexity meaning storage, the amount of time, or other necessary resources depends on the algorithm utilized to sort each bucket, the number of buckets to use, and whether the input is consistently distributed.

Bucket sort works as follows:

a) Set up an array of originally empty buckets

b) Scatter the original array, placing each object in its corresponding bucket

c) Sort each non-empty bucket.

d) Gather the buckets in order and put all elements back into its original array.

Corporate IT demands ways to classify the vastness of big data, distribute it, and remove the unnecessary ones. Although there are various historical IT practices that can be performed to manage big data, there are differences between traditional transactional data and big data that cannot be disregarded. The clearest difference between big data and transactional data is that transactional data is structured into fixed record lengths, which makes the data easier to manage. However, big data arises in all sizes and shapes.

3.2.1 Recognize the various sizes of big data

Big data can be unstable and unpredictable (Mary Shacklett, (November 18, 2015)) which is why IT should search for new ways to classify the data for purposes of management. One of these approaches is defining different data buckets into which data is located for classification. These various buckets are defined by the size of the data that each bucket carries, as well as by the end user groups for which this data is processed and eventually sent to.

One method to describe big data buckets is by the size of the data.

- Big data is utilized for historical analytics over wide durations in a healthcare companies, hospitality industry, public service agencies, and retail businesses. Big Data runs in a batch mode on a big data engine like for example Hadoop; an open source distributed processing framework that manages data storage and processing for big data applications running in clustered systems.

  The data for jobs of this nature exist in big data lakes (a storage repository that holds vast amounts of raw_data in its initial format until it is needed), and it can take several hours to distribute and process.
- Medium-size data can be located in a data mart subset of data.
- Small data originates in the form of high-volume data snippets that flow through data conduits rapidly, and that are directly actionable, such as responding and monitoring the temperature readings of a thermostat at a remote location.

  By comprehending the various sizes of big data, and where they are possibly needed in the enterprise, IT is in a healthier position to measure the storage and processing resources needed for this data. This information aids also to form overall IT investment decisions and architecture, as well as choices on what big data to outsource and which to preserve internally.

3.2.2 Best practices

1: Understand payloads, sizes of the data, and the stakeholders for each classification.

The features of data payloads will probably define the types of IT investments in processing and storage made.

2: Determine what can be outsourced and what must be maintained internally

High data security must preferably be reserved in enterprise walls, however if huge chunks of data needs to be stored and accessed in the future and don’t transmit important security requirements, then cloud-based storage service should be taken into consideration.

3: Define the enterprise business cases

Without a business case, it is hard to validate IT investments and the return on them or to charge end users for services.

3.2.3 Elaboration on Data Buckets

Basically, the data is not only split into even parts to enhance the search, but is also being divided into buckets types.

First, the data incoming from users is to be added to the big data environment that will be assessed by one of the preprocessing algorithms explained above. In addition, the load balancer will split the data into multiple adjacent and similar buckets having the same amounts of data from the same type into multiple buckets of the same type.

Second, while these methods are in action, a field will be added to each inserted record on the database tier to identify the data in which bucket it resides along with multiple parameters related to the type, length, synonyms…

Finally, when a pull request is issued, the system will handle this request with the preprocessors and will search only the buckets that are entitled and configured to hold such data. For example, a user is uploading an image of type jpeg containing a university building, and giving this image a title of “University of Notre Dame Louaize, Zouk Mosbeh big entrance in HD”.

The preprocessor will process this image and split it into two entities:

a) Image
b) Text explaining the image.

The image preprocessor will handle the image and will generate a field holding: the physical name of the image, the size of the image and the type of the image. Once the preprocessor knows the type and size of image, it will search for the uneven bucket that falls into that category for example (jpeg, 500KB) and index the image with a specified ID under the required bucket. Let us suppose that the system is splitting the bucket based on type and sub-buckets of that type based on the size. This will end up having a system containing as example multiple jpeg buckets split into multiple sub categories 0->500, 501->1000…. However, if the data is not related to any bucket, then it will create a new bucket that fits the parameters of this image.

On the other hand, the text preprocessor will handle the title of the image, split it into words, phrases of different lengths, index each one of them alongside the indexing of the whole sentence and will relate all of these fragments to the original images. The words will be put into buckets based on the leading character; the phrases based on the word count and will add also a field for these inputs to identify the exact location of these words inside the bucket ecosystem.

When a request emerges from any client asking for university, NDU, Notre Dame, Big Entrance, Jpeg images of entrance…. The system will take the query, split it into words, phrases and start matching them with their synonyms to the existing data. Each match will return the result alongside the bucket where it resides to retrieve rapidly the actual data from the image preprocessors and buckets or from the text preprocessors and buckets. The beauty of this method is that when a match is found, the search will be no longer needed since the relation was already created when inserting the data. Thus, if the university is found, the images of the university are directly accessed and then the data will be filtered based on the other words existing in the search query until a group of image is satisfying the search query as a whole and that result will be returned to the client.

3.2.4 PseudoCode

```
function bucketSort(array, k) is
  buckets ← new array of k empty lists
  M ← the maximum key value in the array
  for i = 1 to length(array) do
    insert array[i] into buckets[floor(array[i] / M * k)]
  for i = 1 to k do
    nextSort(buckets[i])
  return the concatenation of buckets[1], ...., buckets[k]
```

The array is to be sorted and k is the number of buckets to use. The maximum key value can be computed in linear time by looking up all the keys once. The floor function must be used to convert a floating number to an integer. The function nextSort is a sorting function used to sort each bucket. Conventionally, insertion sort would be used, but other algorithm could be used as well. Using bucketSort itself as nextSort produces a relative of radix sort; in particular, the case n = 2 corresponds to quicksort (although potentially with poor pivot choices).

Bucket sort is generally valuable when the input is distributed uniformly over a certain range. When the input holds numerous keys close to each other (clustering), then those elements are likely to be placed in the same bucket.

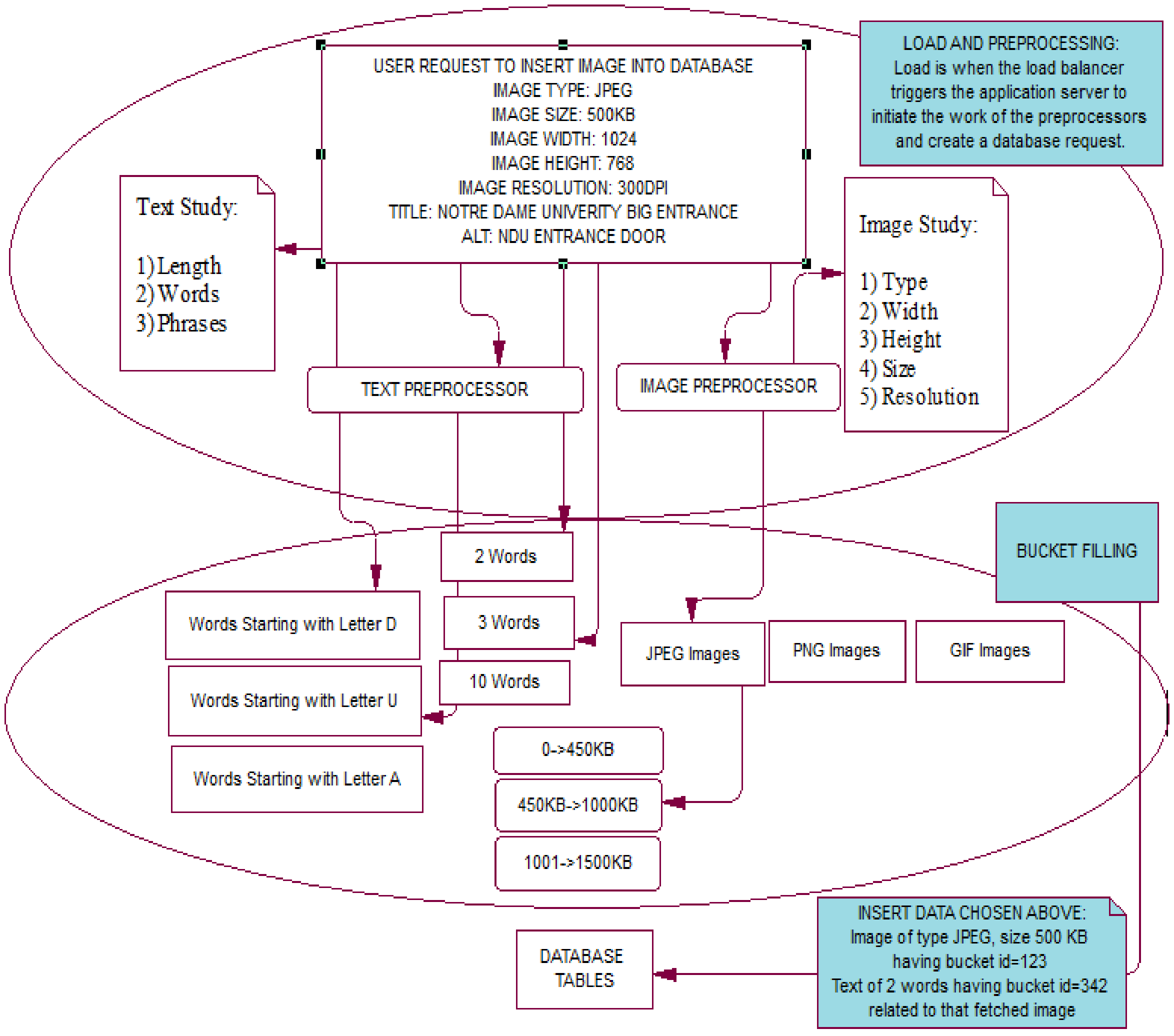

USER REQUEST TO INSERT IMAGE INTO DATABASE
IMAGE TYPE: JPEG
IMAGE SIZE: 500KB
IMAGE WIDTH: 1024
IMAGE HEIGHT: 768
IMAGE RESOLUTION: 300DPI
TITLE: NOTRE DAME UNIVERITY BIG ENTRANCE
ALT: NDU ENTRANCE DOOR
LOAD AND PREPROCESSING: Load is when the load balancer triggers the application server to initiate the work of the preprocessors and create a database request.
Text Study:
1) Length
2) Words
3) Phrases
Image Study:
1) Type
2) Width
3) Height
4) Size
5) Resolution
TEXT PREPROCESSOR
IMAGE PREPROCESSOR
BUCKET FILLING
2 Words
3 Words
10 Words
Words Starting with Letter D
Words Starting with Letter U
Words Starting with Letter A
JPEG Images
PNG Images
GIF Images
0->450KB
450KB->1000KB
1001->1500KB
DATABASE TABLES
INSERT DATA CHOSEN ABOVE: Image of type JPEG, size 500 KB having bucket id=123 Text of 2 words having bucket id=342 related to that fetched image

Figure 25: Schema of Data Buckets

3.2.5 Data Buckets Demo implementation Example

Development environment:

- Eclipse
- JDK 1.8 or greater,
- -Ant 1.8.2+ (We installed the
- Ivy 2.2.0

Frameworks:

- LUCENE: lucene-7.3.1

In our example, simple text files serve as a body for the search.

Lucene used and Defined retrieval and searching Functions:

- IndexSearcher: Create Index
- QueryParser: parse information or translate information into machine logic.
- stopAnalyzer: build an analyser with the stop words from a given sets based on delimiters.

To perform a specific search request:

- Get Document type field (image, media, text)
- Read from tags the related criteria (width, size, type, resolution)

Classes shown are console based with no interface. Those are demo codes to show the basic functions.

Creating Buckets

In this example, buckets will be created according to types. Then subcategories will be created based on this type (parent/child relation)

Showing below is the manual way to create a bucket for testing:

1) Go to the Content Editor, then the content tree, then create a content item named "TEXT_BUCK"
2) In the content tree, select the "TEXT_BUCK", go to "Home Tab", click "Edit" to lock the item.
3) Click Configure tab, go to Buckets group, convert new item into an item bucket.

Making Content Items Bucketable:

1) Category/Subcategory
2) Go to Template Manager, sleect item <IMG_BUCK> (parent) item.
3) Select the _Standard Values item. {(0-450), {451-1000},(1001-1500)},Unit = KB
4) Go to Item Buckets section.
5) Check Lock Child Relationship box.

Extending requires using SOLR   if HTTTP API is more preferable and using higher level of programming.

If we took text query implicitly or explicitly, the select query from data bucket will be executed. In the Related Text Data Bucket through the documents using documents delimiters.

Example:

FROM ~/DATA_BUCKET_DEMO/TEXT_BUCK/ExampleDocumentDemo1.csv

Then take the stream in each document and analyze it according to the processes already implemented. It will depend on the Stream info, stopAnalyser the tokenization used.

Example of Token stream info:

```
public void EXMPDisTokenStreamInfo() {
                String text = Data Bucket Example HERE";
                AnalyzerTool analyzerTool = new AnalyzerTool();
                Analyzer st = new StandardAnalyzer(Version.LUCENE_35);
                Analyzer sa = new StopAnalyzer(Version.LUCENE_35);
                Analyzer sia = new SimpleAnalyzer(Version.LUCENE_35);
                Analyzer ws = new WhitespaceAnalyzer(Version.LUCENE_35);
                analyzerTool.disTokenStreamInfo(text, st);
                logger.info("--------Data--------");
                analyzerTool.disTokenStreamInfo(text, sa);
                logger.info("--------Bucket--------");
```

```
                              analyzerTool.disTokenStreamInfo(text, sia);
                              logger.info("--------Example --------");
                              analyzerTool.disTokenStreamInfo(text, ws);
                              logger.info("--------HERE--------");
```

Lucene Constants in Demo:

```
public class LuceneConstants {
  public static final String CONTENTS = "contents";
  public static final String FILE_NAME = "filename";
  public static final String FILE_PATH = "filepath";
  public static final int MAX_SEARCH = 10;}
```

Note: this class is needed for the basic start.

Lucene Filter Demo code:

```
import java.io.FileFilter;
import java.io.FileReader;
import java.io.IOException;
import java.nio.file.Paths;

import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.TextField;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;



  public class TextFileFilter implements FileFilter {
/* */
  @Override
  public boolean accept(File pathname) {
    return pathname.getName().toLowerCase().endsWith(".txt");
  }
}
```

Lucene Searcher Demo code:

```
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.queryparser.classic.ParseException;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;

public class Searcher {

   IndexSearcher indexSearcher;
   QueryParser queryParser;
   Query query;

   public Searcher(String indexDirectoryPath)
      throws IOException {
      Directory indexDirectory =
         FSDirectory.open(Paths.get(indexDirectoryPath));
      IndexReader reader = DirectoryReader.open(indexDirectory);
      indexSearcher = new IndexSearcher(reader);
      queryParser = new QueryParser(LuceneConstants.CONTENTS,
         new StandardAnalyzer());
   }

   public TopDocs search( String searchQuery)
      throws IOException, ParseException {
      query = queryParser.parse(searchQuery);
      return indexSearcher.search(query, LuceneConstants.MAX_SEARCH);
   }

   public Document getDocument(ScoreDoc scoreDoc)
      throws CorruptIndexException, IOException {
      return indexSearcher.doc(scoreDoc.doc);
   }

}
```

Lucene Test Demo code:

```
import java.io.IOException;

import org.apache.lucene.document.Document;
import org.apache.lucene.queryparser.classic.ParseException;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopDocs;

public class LuceneTester {

   String indexDir = "/home/Index/";
   String dataDir = "/home/Data/";
   Indexer indexer;
   Searcher searcher;

   public static void main(String[] args) {
      LuceneTester tester;
      try {
         tester = new LuceneTester();
         tester.createIndex();
         tester.search("YourSearchTerm");
      } catch (IOException e) {
         e.printStackTrace();
      } catch (ParseException e) {
         e.printStackTrace();
      }
   }

   private void createIndex() throws IOException {
      indexer = new Indexer(indexDir);
      int numIndexed;
      long startTime = System.currentTimeMillis();
      numIndexed = indexer.createIndex(dataDir, new TextFileFilter());
      long endTime = System.currentTimeMillis();
      indexer.close();
      System.out.println(numIndexed+" File indexed, time taken: "
         +(endTime-startTime)+" ms");
   }

   private void search(String searchQuery) throws IOException, ParseException {
      searcher = new Searcher(indexDir);
      long startTime = System.currentTimeMillis();
      TopDocs hits = searcher.search(searchQuery);
      long endTime = System.currentTimeMillis();

      System.out.println(hits.totalHits +
         " documents found. Time :" + (endTime - startTime));
      for(ScoreDoc scoreDoc : hits.scoreDocs) {
         Document doc = searcher.getDocument(scoreDoc);
            System.out.println("File: "
            + doc.get(LuceneConstants.FILE_PATH));
      }
   }
}
```

Conclusion

As a conclusion, big data technology is certainly the answer for the huge need to store and manipulate current application data load.

In addition, big data has already solved major performance issues comparing to traditional relational databases where the data is centralized and benefit from a singular server farm. The data volume is scaling much faster than the computer resources, that’s why merging between scale up and scale out is a must to cope with the increasing data volume.

Finally, bucketing is used to split your table into various files that can be read without doing a full table scan. The beauty of using buckets is that when a match is found, the search will be no longer needed since the relation was already created when inserting the data. Buckets provide portion of partitions using hashing algorithms. Buckets allow effective retrieval of sample data, efficient division of work and of join operations.

4. References

Isaacson, C. (2015). Understanding Big Data Scalability: Big Data Scalability Series, Part I. [online] O'Reilly | Safari. Available at: https://www.safaribooksonline.com/library/view/understanding-big-data/9780133599121/pref01.html [Accessed 2015].

Anon, (2019). [online] Available at: https://www.researchgate.net/figure/Six-Vs-of-big-data-value-volume-velocity-variety-veracity-and-variability-which_fig2_280124446 [Accessed Feb. 2016].

Techopedia.com. (n.d.). *What is Big Data? - Definition from Techopedia*. [online] Available at: https://www.techopedia.com/definition/27745/big-data.

Ijarcce.com. (2017). [online] Available at: https://ijarcce.com/upload/2017/si/ICACTRP-17/IJARCCE-ICACTRP%2029.pdf [Accessed 2 Feb. 2017].Sonawane, A. (2009). Using Apache Lucene to search text. [online] Ibm.com. Available at: https://www.ibm.com/developerworks/library/os-apache-lucenesearch/index.html [Accessed 18 Aug. 2009].

Inpressco.com. (2014). [online] Available at: http://inpressco.com/wp-content/uploads/2014/09/Paper373286-3289.pdf [Accessed 1 Oct. 2014].

Hal.archives-ouvertes.fr. (2015). [online] Available at: https://hal.archives-ouvertes.fr/hal-01213310/document [Accessed 1 May 2015].

XenonStack. (2018). Stateful and Stateless Application Development Solutions - XenonStack. [online] Available at: https://www.xenonstack.com/insights/what-are-stateful-and-stateless-applications/ [Accessed 11 Dec. 2018].

Lohr, S. (2012). Opinion | Big Data's Impact in the World. [online] Nytimes.com. Available at: https://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html [Accessed 12 Feb. 2012].

Patil, D. (2011). NPR Choice page. [online] Npr.org. Available at: http://www.npr.org/2011/11/30/142893065/the-¬ -search-for -analysts
¬ -make -sense-of -big -data [Accessed 30 Nov. 2011].

En.wikipedia.org. (2019). Bucket sort. [online] Available at: https://en.wikipedia.org/wiki/Bucket_sort [Accessed 29 Apr. 2019].

NGINX. (n.d.). *What Is Load Balancing? How Load Balancers Work*. [online] Available at: https://www.nginx.com/resources/glossary/load-balancing/.

Charnock, E. (2011). NPR Choice page. [online] Npr.org. Available at: https://www.npr.org/2011/11/29/142521910/the-digital-breadcrumbs-that-lead-to-big-data [Accessed 29 Nov. 2011].

Whiting, B. (n.d.). What is Big Data Analytics? - Definition & Examples | Study.com. [online] Study.com. Available at: https://study.com/academy/lesson/what-is-big-data-analytics-definition-examples.html.

Dijcks, J. (2013). [online] Oracle.com. Available at: http://www.oracle.com/us/products/database/big-data-for-enterprise-519135.pdf [Accessed 1 Jun. 2013].

Rouse, m. (2019). What is Hadoop? - Definition from WhatIs.com. [online] SearchDataManagement. Available at: https://searchdatamanagement.techtarget.com/definition/Hadoop [Accessed May 2019].

Heidelberg, S. (2014). Big Data. [online] Google Books. Available at: https://books.google.com.lb/books?id=0wwqBAAAQBAJ&pg=PA9&lpg=PA9&dq=http://www.economist.com/blogs/dailychart/2011/11/big-%C2%AC%E2%80%90data-%C2%AC%E2%80%900&source=bl&ots=Hk3kwQjnA8&sig=ACfU3U14B3M_7VguAX5kjqovWHW9cK0JsQ&hl=en&sa=X&ved=2ahUKEwiplILp0KnhAhVF1-AKHQFyCVgQ6AEwAXoECAgQAQ#v=onepage&q=http%3A%2F%2Fwww.economist.com%2Fblogs%2Fdailychart%2F2011%2F11%2Fbig-%C2%AC%E2%80%90data-%C2%AC%E2%80%900&f=false [Accessed 20 Nov. 2014].

Shacklett, M. (2015). 3 best practices for defining big data buckets. [online] TechRepublic. Available at: https://www.techrepublic.com/article/3-best-practices-for-defining-big-data-buckets/ [Accessed 18 Nov. 2015]

Tuomainen, R. (2016). [online] Cs.helsinki.fi. Available at: https://www.cs.helsinki.fi/u/jilu/paper/tuomainen.pdf [Accessed 31 Mar. 2016].

Kervizic, J. (2017). Overview of efficiency concepts in Big Data Engineering. [online] Medium. Available at: https://medium.com/analytics-and-data/overview-of-efficiency-concepts-in-big-data-engineering-418995f5f992 [Accessed 7 Mar. 2017].

Garulli, L. (2010). Bucket/Key/Value Paradigm as a Replacement For Relational DBMS - DZone Database. [online] dzone.com. Available at: https://dzone.com/articles/bucketkeyvalue-paradigm [Accessed 8 Apr. 2010].

Stratoscale.com. (n.d.). [online] Available at: https://www.stratoscale.com/blog/cloud/software-load-balancers-5-key-capabilities/.

Nyo Aye, K. (2011). [online] Available at: https://www.researchgate.net/figure/Functional-model-of-multimedia-search-engine_fig3_258550480 [Accessed 1 Dec. 2011].

Nyo Aye, K. (2011). [online] Available at: https://www.researchgate.net/figure/Unstructured-data-indexing-and-retrieval-framework_fig4_258550480 [Accessed 1 Dec. 2011].

Sathishtk.com. (2016). Big Data – TK's blog. [online] Available at: http://sathishtk.com/blog/category/big-data/ [Accessed 25 Jul. 2016].

Codopia. (2017). Capacity Planning and Scaling the Azure Function Apps. [online] Available at: https://codopia.wordpress.com/2017/10/28/capacity-planning-and-scaling-the-azure-function-apps/ [Accessed 28 Oct. 2017].

Avi Networks. (2011). What is Round Robin Load Balancing? Definition & FAQs | Avi Networks. [online] Available at: https://avinetworks.com/glossary/round-robin-load-balancing/ [Accessed 6 Apr. 2011].

Incapsula.com. (2011). [online] Available at: https://www.incapsula.com/load-balancing/sticky-session-persistence-and-cookies.html [Accessed 6 Apr. 2011].

Google.com.lb. (n.d.). *Image: Windows 10 Your Computer is Low on Memory [Solved] - Driver Easy*. [online] Available at: https://www.google.com.lb/imgres?imgurl=https%3A%2F%2Fimages.drivereasy.com%2Fwp-content%2Fuploads%2F2017%2F03%2F1-7.png&imgrefurl=https%3A%2F%2Fwww.drivereasy.com%2Fknowledge%2Fwindows-10-computer-low-memory%2F&docid=j22tMW9q6_ynyM&tbnid=utRlD6vOBZlwjM%3A&vet=10ahUKEwj6k9jzxZXiAhVvx4UKHTXeBHkQMwg8KAAwAA..i&w=357&h=184&bih=606&biw=1366&q=low%20memory&ved=0ahUKEwj6k9jzxZXiAhVvx4UKHTXeBHkQMwg8KAAwAA&iact=mrc&uact=8.

Google.com.lb. (n.d.). *Image: How To Fix Cisvc.exe High CPU Usage | Windows EXE Errors*. [online] Available at: https://www.google.com.lb/imgres?imgurl=http%3A%2F%2Fwindows-exe-errors.com%2Fwp-content%2Fuploads%2Fhigh-cpu-usage.gif&imgrefurl=http%3A%2F%2Fwindows-exe-errors.com%2Fhow-to-fix-cisvc-exe-high-cpu-usage%2F&docid=3tAnWnn-6EhpPM&tbnid=Y8Sh3LD-oarSTM%3A&vet=10ahUKEwiDp6efxZXiAhU58uAKHSWIDKEQMwhAKAUwBQ..i&w=465&h=482&bih=606&biw=1366&q=high%20cpu%20usage&ved=0ahUKEwiDp6efxZXiAhU58uAKHSWIDKEQMwhAKAUwBQ&iact=mrc&uact=8.

Google.com.lb. (n.d.). *Image: Troubleshoot 100% Disk Usage in Windows 10*. [online] Available at: https://www.google.com.lb/imgres?imgurl=https%3A%2F%2Fwww.online-tech-tips.com%2Fwp-content%2Fuploads%2F2016%2F09%2Fhigh-disk-usage-2.png&imgrefurl=https%3A%2F%2Fwww.online-tech-tips.com%2Fwindows-10%2Ftroubleshoot-100-disk-usage-windows-10%2F&docid=ccW5Ef-ReMVOgM&tbnid=nJI2PgB6m14wiM%3A&vet=10ahUKEwiDhs7gxZXiAhUEzYUKHXOFArwQMwhAKAAwAA..i&w=581&h=291&bih=606&biw=1366&q=h

igh%20diskusage&ved=0ahUKEwiDhs7gxZXiAhUEzYUKHXOFArwQMwhAKAAwAA&iact=mrc&uact=8.

Google.com.lb. (n.d.). *data acquisition in big data - Google Search*. [online] Available at: https://www.google.com.lb/search?biw=1366&bih=651&tbm=isch&sa=1&ei=PE_YXNS1PJCca6yurMAG&q=data+acquisition+in+big+data&oq=data+acquisition+in+big+data&gs_l=img.3..0i24.5739.7734..8438...0.0..0.162.1586.0j12......0....1..gws-wiz-img.......0j0i67j0i8i30.dviO1-BDuO8#imgrc=llAdcb6GoGmUzM:.

Google.com.lb. (n.d.). *load balancer in big data - Google Search*. [online] Available at: https://www.google.com.lb/search?biw=1366&bih=651&tbm=isch&sa=1&ei=R0_YXPetDoyelwSI6aiYBA&q=load+balancer+in+big+data&oq=load+balancer+in+big+data&gs_l=img.3...577300.580174..580817...0.0..0.164.2041.0j15......0....1..gws-wiz-img.z5UFrvral2Y#imgrc=avLgAVurBjrWYM:.

Cross, A. (2018). *The Importance of Scalability in Big Data Processing – NGDATA*. [online] NGDATA. Available at: https://www.ngdata.com/the-importance-of-scalability-in-big-data-processing/ [Accessed 20 Aug. 2018].

Stratoscale.com. (n.d.). [online] Available at: https://www.stratoscale.com/blog/cloud/software-load-balancers-5-key-capabilities/.

Google.com.lb. (n.d.). *application server tier database tier - Google Search*. [online] Available at: https://www.google.com.lb/search?biw=1366&bih=606&tbm=isch&sa=1&ei=WWXYXIPTOMjeasOOjgg&q=application+server+tier+database+tier&oq=application+server+tier+database+tier&gs_l=img.3...143062.145104..145232...0.0..0.162.1629.0j12......0....1..gws-wiz-img.wcBGnfU1lic#imgrc=nBTOrtuRckr6YM:.

https://www.google.com.lb/search?q=unstructured+data+in+big+data&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjQqrfN0pbiAhUFxYUKHf6BA9MQ_AUIDigB&biw=1366&bih=606#imgrc=O9ybUwjojv4nmM:

Google.com. (n.d.). *big data - Google Search*. [online] Available at: https://www.google.com/search?q=big+data&safe=active&rlz=1C1GCEB_enLB802LB802&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiyxOrRz53iAhWsUBUIHeWmBNcQ_AUIDigB&biw=1600&bih=789#imgrc=n-ccgrGfyTv6wM:.

Google.com. (n.d.). *Image: What is vertical scalability (scaling up)? - Definition from ...*. [online] Available at: https://www.google.com/imgres?imgurl=https%3A%2F%2Fitknowledgeexchange.techtarget.com%2Foverheard%2Ffiles%2F2016%2F01%2Fvertical-scalability.png&imgrefurl=https%3A%2F%2Fsearchcio.techtarget.com%2Fdefinition%2Fvertical-scalability&docid=ocWuprDWLMySeM&tbnid=6H6a8vdotsYyIM%3A&vet=10a

hUKEwiI7ISJ153iAhUEBWMBHWOBDnUQMwg7KAAwAA..i&w=516&h=244&safe=active&bih=740&biw=1600&q=scalability%20definition%20in%20big%20data%20image&ved=0ahUKEwiI7ISJ153iAhUEBWMBHWOBDnUQMwg7KAAwAA&iact=mrc&uact=8.

Google.com.lb. (n.d.). *table about big data - Google Search*. [online] Available at: https://www.google.com.lb/search?q=table+about+big+data&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiEwZ7ppp_iAhWRnhQKHVmSBT8Q_AUIDigB&biw=1366&bih=651#imgrc=Teyh9m9aA-mBqM:.

Google.com.lb. (n.d.). *Google Image Result for https://www.datamation.com/imagesvr_ce/8800/structured-unstructured.jpg*. [online] Available at: https://www.google.com.lb/imgres?imgurl=https%3A%2F%2Fwww.datamation.com%2Fimagesvr_ce%2F8800%2Fstructured-unstructured.jpg&imgrefurl=https%3A%2F%2Fwww.datamation.com%2Fbig-data%2Fstructured-vs-unstructured-data.html&docid=BCjgltpCz2DwnM&tbnid=Bjtb3erP3ZNruM%3A&vet=10ahUKEwjutYTRrp_iAhXnDGMBHarNDJYQMwhJKAwwDA..i&w=500&h=350&bih=651&biw=1366&q=big%20data%20versus%20relational%20database&ved=0ahUKEwjutYTRrp_iAhXnDGMBHarNDJYQMwhJKAwwDA&iact=mrc&uact=8.

Google.com.lb. (n.d.). *graphs heterogeneity versus homegenity - Google Search*. [online] Available at: https://www.google.com.lb/search?q=graphs+heterogeneity+versus+homegenity&tbm=isch&tbs=rimg:CfgKouVZbPK5Ijg_1T66wXhxyrKnud8Vk7eGcYzPKsfhHBmCse87tGzgLYkYabM9kR_11-MSfiqwUCfwsxOM4S_1VTlHCoSCT9PrrBeHHKsEXZ_1YmjenZW8KhIJqe53xWTt4ZwROY8D6SSPmnAqEgljM8qx-EcGYBHkEGA6VbUfqioSCax7zu0bOAtiEbRDHFUYtxbZKhIJRhpsz2RH_1X4RbIt8Jlv2NX4qEgkxJ-KrBQJ_1CxFoHTabR3A18CoSCTE4zhL9VOUcERJ2bkcNYVSc&tbo=u&sa=X&ved=2ahUKEwi009HB05_iAhUHohQKHWLmAIwQ9C96BAgBEBg&biw=1242&bih=592&dpr=1.1#imgrc=MSfiqwUCfwuohM:.

Google.com.lb. (n.d.). *scalability of data big data - Google Search*. [online] Available at: https://www.google.com.lb/search?biw=1242&bih=592&tbm=isch&sa=1&ei=xyHdXNyXKL6EjLsPoZGI6AE&q=scalability+of+data+big+data&oq=scalability+of+data+big+data&gs_l=img.3...468933.477947..478426...7.0..0.281.5161.0j30j3......0....1..gws-wiz-img.......0j0i67j0i8i30j0i24.y6npcImNi28#imgrc=NM_FGJCayhmqVM:.

Antonopoulos, C. (n.d.). *The Importance of Data Quality Timeliness - Measured Results Marketing*. [online] Measured Results Marketing. Available at: https://www.measuredresultsmarketing.com/importance-data-quality-timeliness/.

Google.com.lb. (n.d.). *complexity of data big data definition - Google Search*. [online] Available at: https://www.google.com.lb/search?biw=1242&bih=551&tbm=isch&sa=1&ei=Dy7dXPWUN8uua_mBt_AJ&q=complexity+of+data+big+data+definition&oq=complexity+of+data+big+data+definition&gs_l=img.3...550729.552552..553168...0.0..0.412.2891.0j2j3j4j1......0....1..gws-wiz-img.gOlVa_zu8BA#imgrc=BW8_i-UWVWmLmM:.

Jain, P., Gyanchandani, M. and Khare, N. (n.d.). *Big data privacy: a technological perspective and review*.

Google.com.lb. (n.d.). *bucket of data - Google Search*. [online] Available at: https://www.google.com.lb/search?biw=1242&bih=551&tbm=isch&sa=1&ei=wT3dXKniFKyWlwSn6LeoCQ&q=bucket+of+data++&oq=bucket+of+data++&gs_l=img.3...194317.201408..201968...1.0..1.175.3437.0j26......0....1..gws-wiz-img.......0i8i7i30.vzioeM_TFG0#imgrc=0l0hQbMAgF7BQM:.