**An Approximate Search Framework for Big Data**

**Abstract:**

In the age of big data, a traditional scanning search pattern is gradually becoming unfit for a satisfying user experience due to its lengthy computing process. In this paper, we propose a sampling-based approximate search framework called Hadoop framework( map reduce), to meet user’s query demand for both accurate and efficient results .In novel frame work is presented to uniformly measure accuracy and efficiency for a big data search service, which enables to work out a feasible searching job. Based on this, we employ the bootstrapping technique to further speed up the search process. Moreover, an incremental sampling strategy is investigated to process homogeneous queries; in addition, the reuse theory of historical results is also studied for the scenario of appending data. Theoretical analyses and experiments on a real-world dataset demonstrate that map reduce algorithm is capable of producing approximate results meeting the preset query requirements with both high accuracy and efficiency.

**Introduction**

The thriving development of big data witnesses the fading of the traditional scanning search pattern, which has posed unprecedented challenges to the existing data processing applications. Obtaining both fast and accurate results represents a crucial desideratum for user experience. Current search systems and computing platforms, however, cannot seek out the exact results by scanning the whole dataset in a timely manner. Therefore, a much more realistic search frame work should provide users with accurate enough and fairly quick approximate answers, rather than dawdling over exact ones.

A number of distributed infrastructures (such as Hadoop) enable developers to easily leverage thousands of computing nodes to perform data-parallel computations [3]; it seems to be a panacea for big data searches to adopt sampling methods(such as bootstrapping and jackknife [4]) in the computing process. Unfortunately, due to the lack of communication approaches between users and the computing platform, there as on ability of a query with both accuracy and efficiency requirements cannot be evaluated beforehand by considering the available computing resource. Therefore, a uniform metric for accuracy and efficiency needs to be established. Moreover, although much work has been done to study sampling theories applied to Map Reduce-oriented systems there still exist many practical problems to be solved in real life scenarios. For example, since big data query is very time/energy-consuming, it is worth studying whether existing answers can be reusable when: *i)* multiple users submit similar queries differing only in accuracy for the same dataset; *ii)*data-holders append new data to the original dataset.

In our work, an approximate search framework, called *Hermes*, is proposed to solve the series of questions presented above. The chief challenge facing *Hermes* is how to quantify user’s multi-dimensional query requirements with a uniform metric. Therefore, we integrate the query accuracy and efficiency into a uniform metric, (*ε, δ*)-approximation. From the view of functions, *Hermes* is divided into three main components :query evaluation module (QEM), approximate query module (AQM), and query maintenance module (QMM). The QEM resolves the communication problems between user and data platform, and works out a feasible search job through comprehensive analysis of platform state, data distribution and available resources. The AQM, as the core of *Hermes*, is designed as a three-tier architecture: an operation layer, an acceleration layer, supporting a quick response mechanism by reusing the results of previous similar queries, and a sampling layer. The QMM exploits an incremental sampling strategy based on the AQM, which greatly reduces the time cost for queries differing only in accuracy. Additionally, the reliability of historical results when appending data is also discussed. Finally, theoretical analyses and experimental results demonstrate that *Hermes* has good performance in terms of accuracy and efficiency.

**LITERATURE SURVEY**

**1.Networking for Big Data: A Survey**

Complementary to the fancy big data applications, networking for big data is an indispensable supporting platform for these applications in practice. This emerging research branch has gained extensive attention from both academia and industry in recent years. In this new territory, researchers are facing many unprecedented theoretical and practical challenges. We are therefore motivated to solicit the latest works in this area, aiming to pave a comprehensive and solid starting ground for interested readers. We firstly clarify the definition of networking for big data based on the cross disciplinary nature and integrated needs of the domain. Secondly, we present the current understanding of big data from different levels, including its formation, networking features, mathematical representations, and the networking technologies. Thirdly, we discuss the challenges and opportunities from various perspectives in this hopeful field. We further summarize the lessons we learnt based on the survey. We humbly hope this work will shed light for forthcoming researchers to further explore the uncharted part of this promising land.

**2. A Novel Performance Evaluation and Optimization Model for Big Data System**

In recent years, the development of Internet enables the rapid growth of global data volume, the arrival of the era of big data has brought great challenges to the traditional computing. Big Data systems, such as hadoop, spark, are becoming important platforms to handle big data, but due to design flaws of big data application itself, and unreasonable distributed framework configuration, the performance of the applications in big data system is difficult to achieve peak speed of computer theory, so how to locate performance bottleneck of big data system and analyze the bottleneck causes is worthy of research. In this paper, a 5-layer performance evaluation model for big data system is proposed, which is are liable basis for performance analysis, and at the same time, a performance optimization model for big data system is also proposed, which can assist performance bottleneck location and bottleneck analysis, and further optimize performance. Based on these two performance models, an event-based performance tool to profile performance data is implemented. Experimental results show that these two performance models are effective for performance evaluation and optimization of big data system, which can improve average running time of big data system by 19%.

**3. Performance Improvement of MapReduce Process by Promoting Deep Data Locality**

MapReduce has been widely used in many data science applications. It has been observed that an excessive data transfer has a negative impact on its performance. To reduce the amount of data transfer, MapReduce utilizes data locality. However, even though the majority of the processing cost occurs in the later stages, data locality has been utilized only in the early stages, which we call Shallow Data Locality (SDL). As a result, the benefit of data locality has not been fully realized. We have explored a new concept called Deep Data Locality (DDL) where the data is pre-arranged to maximize the locality in the later stages. Toward achieving stronger DDL, we introduce a new block placement paradigm called Limited Node Block Placement Policy (LNBPP). Under the conventional default block placement policy (DBPP), data blocks are randomly placed on any available slave nodes, requiring a copy of RLM (Rack-Local Map) blocks. On the other hand, LNBPP places the blocks in a way to avoid RLMs, reducing the block copying time. The containers without RLM have a more consistent execution time, and when assigned to individual cores on a multi core node, they finish a job faster collectively than the containers under DBPP. LNBPP also rearranges the blocks into a smaller number of nodes (hence Limited Node) and reduces the data transfer time between nodes. These strategies bring a significant performance improvement in Map and Shuffle. Our test result shows that the execution times of Map and Shuffle have been improved by up to 33% and 44%respectively. In this paper, we describe the Map Reduce workflow in Hadoop with a simple computational model and introduce the current research directions in each step. We analyze the block placement status and RLM locations in DBPP with the customer review data from Trip Advisor and measure the performances by executing the Terasort Benchmark with various sizes of data. We then compare the performances of LNBPP with DBPP.

**4. Performance Evaluation of New SQL Databases**

For over forty years, relational databases have been the leading model for data storage, retrieval and management. However, due to increasing needs for scalability and performance, alternative systems have emerged, namely New SQL technology. New SQL is a class of modern relational database management systems(RDBMS) that provide the same scalable performance of No SQL systems for online transaction processing (OLTP)read-write workloads while still maintaining the ACID guarantees of a traditional database system. The rising interest in New SQL technology, over the last few years resulted in an increasing number of evaluations and comparisons among competing New SQL technologies. Some of the New SQL databases currently used in various popular web applications are Volt DB, Google Spanner, MemSQL, SAP HANA, Nuo DB, and TokuDB. This work is trying to comment on the various New SQL database systems which describes benefits, characteristics, and classification of New SQL databases for online transaction processing (OLTP) for Big data management. It also provides the list of popular New SQL databases in categorized tables. This dissertation work mainly covers evaluation comparison between four NewSQL databases: NuoDB, VoltDB, MemSQL, and Cockroach DB on the basis of various parameters like read latency, write latency, update latency, and execution time. The experiments do not cover only performance (latency and execution time) but also focus on ease of use and flexibility of the used New SQL databases.

**5. A Distributed Approach for Top-k Star Queries on Massive Information Networks**

Massive information networks, such as the knowledge graph by Google, contain billions of labeled entities. Star queries, which aim to identify an entity, given a set of related entities, are common on such networks. Answering star queries can be modeled as a graph pattern matching problem. Traditional approaches apply graph indices to accelerate the query processing. Unfortunately, it is so costly that it is nearly infeasible to build indices on billion node graphs since the time or storage complexity of most indexing techniques is super-linear to the graph size. In this paper, we propose an algorithm to identify the top-k best answers for a star query. Instead of using expensive indices, our algorithm utilizes novel bounding techniques to derive the top-k best answers efficiently. Further, the algorithm can be implemented in a distributed manner scaling to billions of entities and hundreds of machines. We demonstrate the effectiveness and the efficiency of our approach through a series of experiments on real-world information networks.

**Existing System:**

A traditional scanning search pattern is gradually becoming unfit for a satisfying user experience due to its lengthy computing process, the processing the real time dataset and analyzing that dataset is very difficult.

**Disadvantages**

* Processing time is more
* Search results is Inaccurate

**Proposed System**

The thriving development of big data witnesses the fading of the traditional

scanning search pattern, which has posed unprecedented challenges to the existing data processing applications. Obtaining both fast and accurate results represents a crucial desideratum for user experience. Current search systems and computing platforms, however, cannot seek out the exact results by scanning the whole dataset in a timely manner. Therefore, a much more realistic search framework should provide users with accurate enough and fairly quick approximate answers, rather than dawdling over exact ones..

**Advantages**

**1.** Accurate results.

**2.** Processing time is less.

**3.** Easily process huge amount of data

Modules:

**1. Data collection :**

In the data collection phase the sensor sensing the data will be collected the data generated will be like twitter events generated data the politician name, time and year in which the events occurred.

**2. Data processing:**

In the data processing phase the data will be processed and send the data to Approximate search network

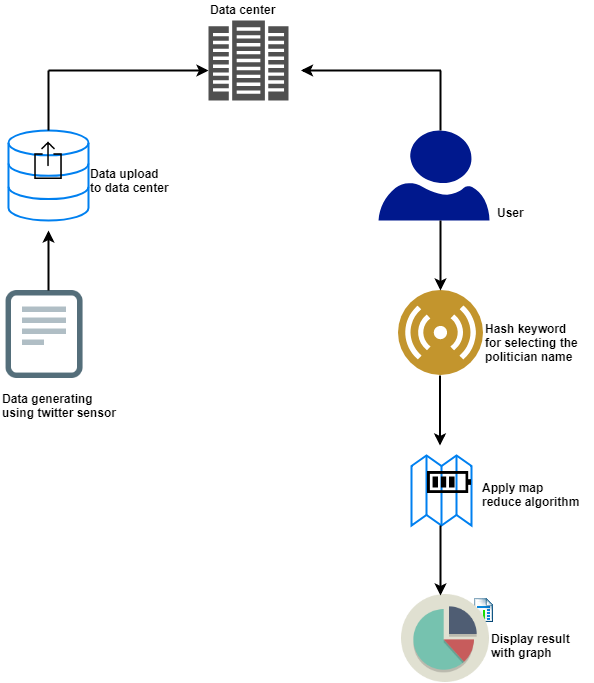
**3. Communication:**

The communication occurred between the sensor and the approximate search network will be established through the socket communication. The user once login to the system search the data by hash keyword i.e.. Politician name in which the particular event occurred and stores the count and display graph by using map-reduce program.

**4. Data storage:**

In the data storage phase approximate network the data will be stored.

**System Architecture**

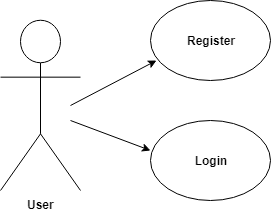
****

**SYSTEM DESIGN**

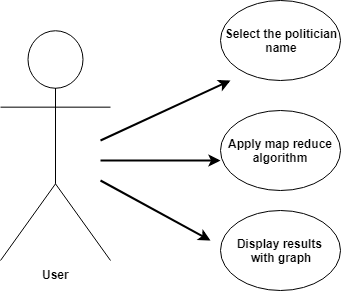
System design is the process of defining the architecture, components, modules, interfaces and [data](http://en.wikipedia.org/wiki/Data) for a [system](http://en.wikipedia.org/wiki/System) to satisfy specified [requirements](http://en.wikipedia.org/wiki/Requirement). One could see it as the application of [systems theory](http://en.wikipedia.org/wiki/Systems_theory) to [product development](http://en.wikipedia.org/wiki/Product_development). There is some overlap with the disciplines of [systems analysis](http://en.wikipedia.org/wiki/Systems_analysis), [systems architecture](http://en.wikipedia.org/wiki/Systems_architecture) and [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering). If the broader topic of [product development](http://en.wikipedia.org/wiki/Product_development) "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing [systems](http://en.wikipedia.org/wiki/System) to satisfy specified [requirements](http://en.wikipedia.org/wiki/Requirement) of the user.

**Use Case Diagram:**

**Use case 1**

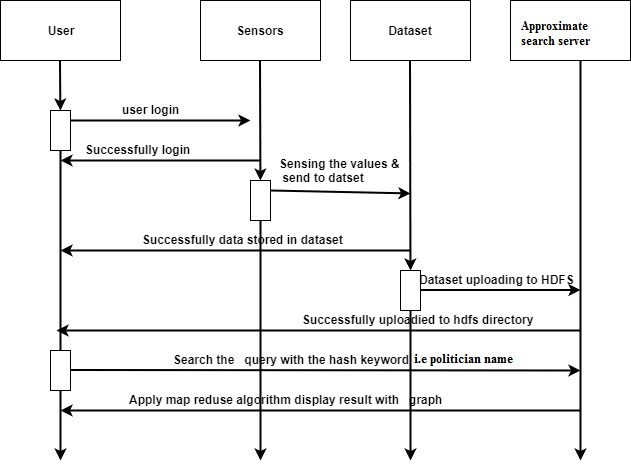


**Use case 2**

****

**Sequence Diagram**

A sequence diagram in a UML is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development.



**Data Flow Diagram**

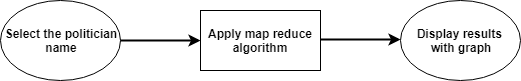
A data flow diagram is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system

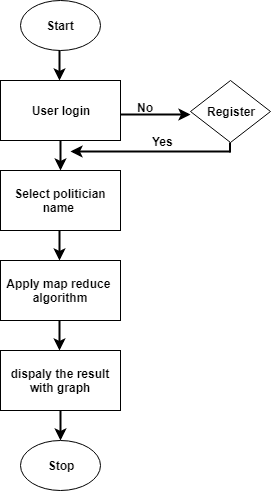
DFD 1



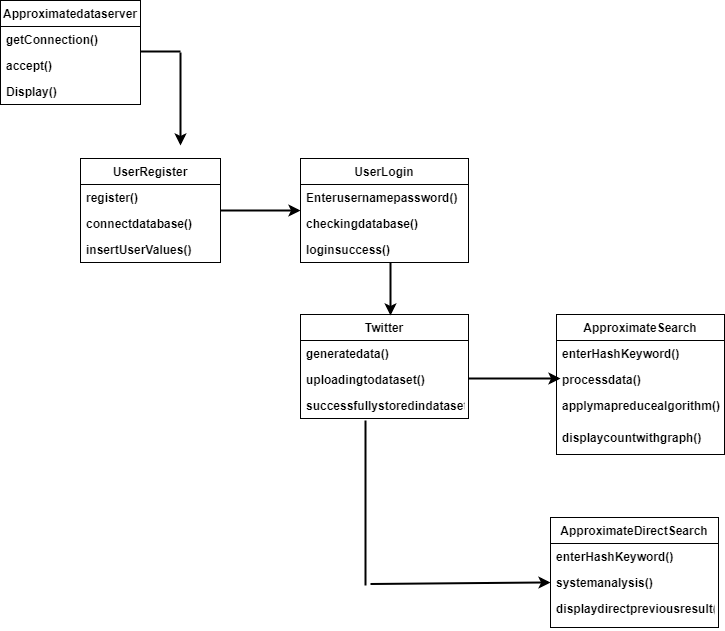
DFD2



**Flow Chart**

****

**Class Diagram:**

****

**SYSTEM REQUIREMENT SPECIFICATION**

To be used efficiently, all [computer software](http://en.wikipedia.org/wiki/Computer_software) needs certain [hardware](http://en.wikipedia.org/wiki/Computer_hardware) components or other software resources to be present on a [computer](http://en.wikipedia.org/wiki/Computer). These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: [minimum](http://en.wikipedia.org/wiki/System_Requirements#Minimum_System_Requirements) and [recommended](http://en.wikipedia.org/wiki/System_Requirements#Recommended_system_requirements). With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

**7.2 Non functional requirements**

Non functional requirements are the functions offered by the system. It includes time constraints and constraints on the development process and standards. The non functional requirements are as follows:

* **Speed:** The system should process the given input into output within appropriate time.
* **Ease of use:** The software should be user friendly. Then the customers can use easily,

so it doesn’t require much training time.

* **Reliability:** The rate of failures should be less then only the system is more reliable
* **Portability**: It should be easy to implement in any system.

**7.2.1 Specific Requirements**

The specific requirements are:

* **User Interfaces:** The external users are the clients. All the clients can use this software for indexing and searching.
* **Hardware Interfaces:** The external hardware interface used for indexing and searching is personal computers of the clients. The PC’s may be laptops with wireless LAN as the internet connections provided will be wireless.
* **Software Interfaces:** The Operating Systems can be any version of Windows.
* **Performance Requirements:** The PC’s used must be atleast Pentium 4 machines so that they can give optimum performance of the product.

**7.3 Software requirements**

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.

These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

* Java1.4 or higher
  + Java Swing – front end
  + Networking-Socket programming
* Windows 98 or higher-Operating System

**7.4 Hardware requirements**

The most common set of requirements defined by any [operating system](http://en.wikipedia.org/wiki/Operating_system) or [software application](http://en.wikipedia.org/wiki/Software_application) is the physical computer resources, also known as [hardware](http://en.wikipedia.org/wiki/Computer_hardware), A hardware requirements list is often accompanied by a [hardware compatibility list](http://en.wikipedia.org/wiki/Hardware_compatibility_list), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

All computer [operating systems](http://en.wikipedia.org/wiki/Operating_system) are designed for a particular [computer architecture](http://en.wikipedia.org/wiki/Computer_architecture). Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture.

The power of the [central processing unit](http://en.wikipedia.org/wiki/Central_processing_unit) (CPU) is a fundamental system requirement for any software. Most software running on [x86 architecture](http://en.wikipedia.org/wiki/X86_architecture) define processing power as the [model](http://en.wikipedia.org/wiki/List_of_microprocessors) and the [clock speed](http://en.wikipedia.org/wiki/Clock_rate) of the CPU. Many other features of a CPU that influence its speed and power, like [bus speed](http://en.wikipedia.org/wiki/Front_side_bus), [cache](http://en.wikipedia.org/wiki/CPU_cache), and [MIPS](http://en.wikipedia.org/wiki/Instructions_per_second) are often ignored. This definition of power is often erroneous, as [AMD](http://en.wikipedia.org/wiki/Advanced_Micro_Devices) [Athlon](http://en.wikipedia.org/wiki/Athlon) and [Intel](http://en.wikipedia.org/wiki/Intel) [Pentium](http://en.wikipedia.org/wiki/Pentium_%28brand%29) CPUs at similar clock speed often have different throughput speeds.

* + - * 10GB HDD(min)
      * 128 MB RAM(min)
      * Pentium P4 Processor 2.8Ghz(min)

**7.5 Overview of technologies**

The technologies used in TARF is described as below:

**7.5.1 History of Java**

Java language was developed by James Gosling and his team at sun Microsystems and released formally in 1995. Its former name is oak. Java Development Kit 1.0 was released in 1996 to popularize java and is freely available on Internet.

**7.5.2 Overview of Java**

Java is loosely based on c++ syntax, and is meant to be Object-Oriented Structure of java is midway between an interpreted and a compiled language. The java compiler into ByteCodes, which are secure and portable across different platforms, compiles Java programs. These byte codes are essentially instructions encapsulated in single type, to what is known as java virtual machine (JVM), which resides in standard browser.

JVM is available for almost all OS. JVM converts these byte codes into machine specific instructions at runtime. Java is actually a platform consisting of three components:

* Java programming language.
* Java library of classes and interfaces.
* Java Virtual Machine

**7.5.3 Features of Java**

* Java is a simple language. It does not make use of pointers, function overloading etc,.
* Java is object-oriented language and supports encapsulation, inheritance, Polymorphism and dynamic binding, but does not support multiple inheritance.
* Everything in java is an object except some primitive data types.
* Java is portable.
* It is an architecture neutral that is java programs once compiled can be executed on any machine that is enabled.
* Java is distributed in its approach and used for Internet programming.
* Java is robust, secured, high performing and dynamic in nature.
* Java supports multithreading. Therefore different parts of the program can be executed at the same time.

**7.6** **Packages**

One of the most innovative features of java is packages. The packages both a naming and a visibility control mechanism we can define classes inside a package that are not accessible by code outside the package. It can define the class members that are only exposed to the other members of the same package. Java uses file system directories to store packages. For example the .class files for any classes you declare to be part of My Package must be stored in the directory called My Package remember that cases significant and directory name must match the package name exactly.

A package hierarchy must be reflected in the file system of your java development system. For example the package declared as -package java.awt.image; needs to be stored in java\awt\image in a windows environment.

**7.10.1 Java.lang package**

The java package, java.lang contains fundamental classes and interfaces closely tied to the language and run time system which includes the root classes that form the class hierarchy, types tied to the language definition, basic exceptions, math functions, threading, security functions as well as some information on the underlying native system.

**7.10.2 Java.util**

Data structures that aggregate objects are the focus of the Java.util package included in the packet is the collections API and organized data structure hierarchy influenced heavily by design pattern consideration.

**7.10.3 Java .security**

It provides the classes and interfaces for security framework. It includes classes that implement an easily configurable, fine grained access control security architecture. The packages also supports a generation and storage of cryptographic public key pairs. Finally this package provides classes that support signed/guarded objects and secure random number generation.

**7.11 Swings**

Swing is a widget toolkit for Java. It’s a part of sun Microsystems Java foundation classes-API for providing graphical user interface for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier abstract window toolkit. Swings provide a native look and feel that emulates look and feel of several look and feel unrelated to the underlying platform. Swings introduced a mechanism that allows the look and feel of every component in an application to be altered without making substantial changes to the application code. The introduction of support for a plugable look and feel allows swing components to emulate for the appearance of native components while still retaining the benefits of platform independence. The above feature also makes it easy to make an application written in swing look very different from native programs if desired.

**Look and feel**

In software design look and feel is used in respect of GUI and comprises of its design, including elements such as colors, shapes, layout and typefaces(the “LOOK”) as well as the behavior of dynamic elements such as button, boxes and menus(the “FEEL”). The term look and feel is used in reference to both software and websites.

# Hadoop :

# What is Big Data

Data which are very large in size is called Big Data. Normally we work on data of size MB(WordDoc ,Excel) or maximum GB(Movies, Codes) but data in Peta bytes i.e. 10^15 byte size is called Big Data. It is stated that almost 90% of today's data has been generated in the past 3 years.

## Sources of Big Data

These data come from many sources like

* **Social networking sites:** Facebook, Google, LinkedIn all these sites generates huge amount of data on a day to day basis as they have billions of users worldwide.
* **E-commerce site:** Sites like Amazon, Flipkart, Alibaba generates huge amount of logs from which users buying trends can be traced.
* **Weather Station:** All the weather station and satellite gives very huge data which are stored and manipulated to forecast weather.
* **Telecom company:** Telecom giants like Airtel, Vodafone study the user trends and accordingly publish their plans and for this they store the data of its million users.
* **Share Market:** Stock exchange across the world generates huge amount of data through its daily transaction.

## 3V's of Big Data

1. **Velocity:** The data is increasing at a very fast rate. It is estimated that the volume of data will double in every 2 years.
2. **Variety:** Now a days data are not stored in rows and column. Data is structured as well as unstructured. Log file, CCTV footage is unstructured data. Data which can be saved in tables are structured data like the transaction data of the bank.
3. **Volume:** The amount of data which we deal with is of very large size of Peta bytes.

## Use case

An e-commerce site XYZ (having 100 million users) wants to offer a gift voucher of 100$ to its top 10 customers who have spent the most in the previous year. Moreover, they want to find the buying trend of these customers so that company can suggest more items related to them.

## Issues

Huge amount of unstructured data which needs to be stored, processed and analyzed.

## Solution

**Storage:** This huge amount of data, Hadoop uses HDFS (Hadoop Distributed File System) which uses commodity hardware to form clusters and store data in a distributed fashion. It works on Write once, read many times principle.

**Processing:** Map Reduce paradigm is applied to data distributed over network to find the required output.

**Analyze:** Pig, Hive can be used to analyze the data.

**Cost:** Hadoop is open source so the cost is no more an issue.

# What is Hadoop

Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume. Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing.It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Moreover it can be scaled up just by adding nodes in the cluster.

## Modules of Hadoop

1. **HDFS:** Hadoop Distributed File System. Google published its paper GFS and on the basis of that HDFS was developed. It states that the files will be broken into blocks and stored in nodes over the distributed architecture.
2. **Yarn:** Yet another Resource Negotiator is used for job scheduling and manage the cluster.
3. **Map Reduce:** This is a framework which helps Java programs to do the parallel computation on data using key value pair. The Map task takes input data and converts it into a data set which can be computed in Key value pair. The output of Map task is consumed by reduce task and then the out of reducer gives the desired result.
4. **Hadoop Common:** These Java libraries are used to start Hadoop and are used by other Hadoop modules.

## Advantages of Hadoop

* **Fast:** In HDFS the data distributed over the cluster and are mapped which helps in faster retrieval. Even the tools to process the data are often on the same servers, thus reducing the processing time. It is able to process terabytes of data in minutes and Peta bytes in hours.
* **Scalable:** Hadoop cluster can be extended by just adding nodes in the cluster.
* **Cost Effective:** Hadoop is open source and uses commodity hardware to store data so it really cost effective as compared to traditional relational database management system.
* **Resilient to failure:** HDFS has the property with which it can replicate data over the network, so if one node is down or some other network failure happens, then Hadoop takes the other copy of data and use it. Normally, data are replicated thrice but the replication factor is configurable.

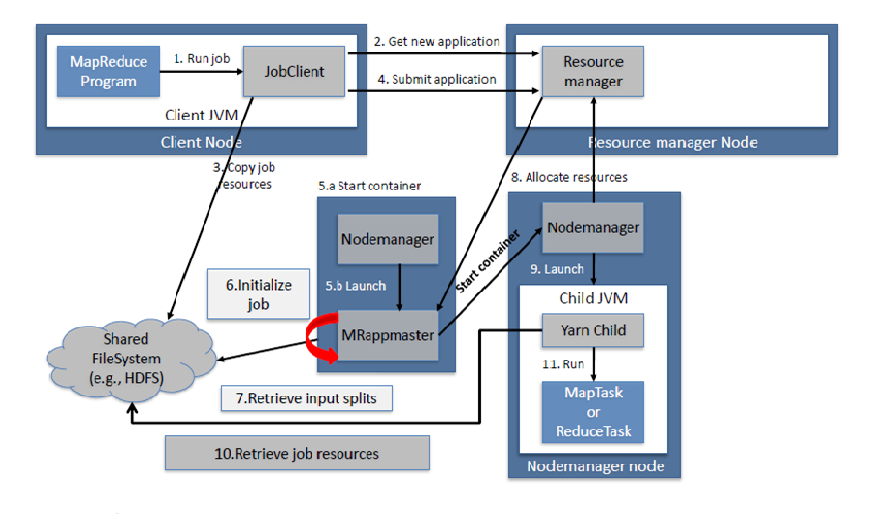
# MapReduce

To take the advantage of parallel processing of Hadoop, the query must be in MapReduce form. The MapReduce is a paradigm which has two phases, the mapper phase and the reducer phase. In the Mapper the input is given in the form of key value pair. The output of the mapper is fed to the reducer as input. The reducer runs only after the mapper is over. The reducer too takes input in key value format and the output of reducer is final output.

## Steps in Map Reduce

* Map takes a data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.
* Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.
* Output of sort and shuffle will be sent to reducer phase. Reducer will perform a defined function on list of values for unique keys and Final output will<key, value> will be stored/displayed.





## How Many Maps

The size of data to be processed decides the number of maps required. For example, we have 1000 MB data and block size is 64 MB then we need 16 mappers.

## Sort and Shuffle

The sort and shuffle occur on the output of mapper and before the reducer.When the mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk.Using the input from each mapper <k2,v2> , we collect all the values for each unique key k2. This output from the shuffle phase in the form of <k2,list(v2)> is sent as input to reducer phase.

**TESTING**

Testing is a critical element which assures quality and effectiveness of the proposed system in (satisfying) meeting its objectives. Testing is done at various stages in the System designing and implementation process with an objective of developing an transparent, flexible and secured system. Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, complies with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested.

* 1. **Test objectives**
* Testing is a process of executing a program with the intent of finding an error.
* A good case is one that has a high probability of finding an undiscovered error.
* A successful test is one that uncovers a yet undiscovered error. If testing is conducted successfully (according to the objectives) it will uncover errors in the software. Testing can't show the absences of defects are present. It can only show that software defects are present.

**9.2 Testing principles**

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

**9.3 Testing design**

Any engineering product can be tested in one of two ways:

**9.3.1 White box Testing**

This testing is also called as glass box testing. Inthis testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function.

it is a test case design method that uses the control structure of the procedural design to derive test cases.

**9.3.2 Black box Testing**

Inthis testing by knowing the internal operation of a product, tests can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Comparison testing

**9.4 Testing strategies**

A software testing strategy provides a road map for the software developer. Testing is a set of activities that can be planned in advanced and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be defined for software engineering process.

**Any software testing strategy should have the following characteristics:**

* 1. Testing begins at the module level and works outward toward the integration of the entire computer based system.
  2. Different testing techniques are appropriate at different points in time.
  3. The developer of the software and an independent test group conducts testing.
  4. Testing and debugging are different activities but debugging must be accommodated in any testing strategy.

**Levels of Testing**

Testing can be done in different levels of SDLC. They are:

**Unit Testing**

The first level of testing is called unit testing. Unit testing verifies on the smallest unit of software designs-the module. The unit test is always white box oriented. In this, different modules are tested against the specifications produced during design for the modules. Unit testing is essentially for verification of the code produced during the coding phase, and hence the goal is to test the internal logic of the modules. It is typically done by the programmer of the module. Due to its close association with coding, the coding phase is frequently called “coding and unit testing.” The unit test can be conducted in parallel for multiple modules.

The Test cases in unit testing are as follows:

**Test Cases**

Table I: Unit Test Case 1

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 1 |
| Description | Approximate search network working properly |
| Input | Approximate search network |
| Expected output | Approximate search network receiving data |
| Actual Result/Remarks | Got the expected output |
| Passed(?) | Yes |

Table II: Unit Test Case 2

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 2 |
| Description | Sensor sending data |
| Input | Twitter |
| Expected output | Data sent successfully |
| Actual Result/Remarks | Got the expected output |
| Passed(?) | Yes |

Table III: Unit Test Case 3

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 3 |
| Description | User registration |
| Input | User register with credentials |
| Expected output | Successful registration |
| Actual Result/Remarks | Got the expected output |
| Passed(?) | Yes |

Table III: Unit Test Case 4

|  |  |
| --- | --- |
| Test Case ID | Unit Test Case 3 |
| Description | User request for data to Approximate search network |
| Input | Sending request |
| Expected output | Successfully receiving data |
| Actual Result/Remarks | Got the expected output |
| Passed(?) | Yes |

**Conclusion:**

an approximate search framework, which divided the search process into three stages: query estimation, searching based on sampling strategies, and performance enhancement. A uniform metric for accuracy and efficiency was established via approximation. Based on the map reduce algorithm could make a feasible search job that users and the data platform. The bootstrapping re sampling technique was employed to speed up the search process within the error bound. To further enhance the performance, an incremental sampling strategy was proposed to deal with homogeneous queries, and the results maintenance theory when appending data was studied as well. Theoretical analysis and experimental results demonstrated that good performance in terms of accuracy and efficiency. Due to the complexity of setting the applicable lower bound of bootstrapping, an empirical value was adopted in the present study. In future work, the global optimal solution to this issue would be explored.

**Reference**

[1] S. Sharma, “Expanded cloud plumes hiding big data ecosystem,” *Future Generation Computer Systems*, vol. 59, no. C, pp. 63–92, 2016.

[2] S. Sharma, U. S. Tim, S. Gadia, R. Shandilya, and S. K. Peddoju,“Classification and comparison of leading NoSQL big data models,” *International Journal of Big Data Intelligence*, vol. 2, no. 3, pp. 201–221, 2015.

[3] S. Yu, M. Liu, W. Dou, X. Liu, and S. Zhou, “Networking for bigdata: A survey,” *IEEE Communications Surveys and Tutorials*, DOI:10.1109/COMST.2016.2610963, 2016.

[4] J. Shao and D. Tu, “The jackknife and bootstrap,” in *Springer series instatistics*. Springer Verlag, 1995.

[5] N. Laptev, K. Zeng, and C. Zaniolo, “Early accurate results for advancedanalytics on mapreduce,” *Proceedings of the Vldb Endowment*, vol. 5,no. 10, pp. 1028–1039, 2012.

[6] K. Zeng, S. Gao, B. Mozafari, and C. Zaniolo, “The analytical bootstrap:a new method for fast error estimation in approximate query processing,”in *Proceedings of the 2014 ACM SIGMOD international conference onManagement of data*, 2014, pp. 277–288.

[7] K. Zeng, S. Gao, J. Gu, B. Mozafari, and C. Zaniolo, “ABS: a system forscalable approximate queries with accuracy guarantees,” in *Proceedingsof the 2014 ACM SIGMOD International Conference on Managementof Data*, 2014, pp. 1067–1070.

[8] H. Herodotou and S. Babu, “Profiling, what-if analysis, and costbasedoptimization of mapreduce programs,” *Proceedings of the VldbEndowment*, vol. 4, pp. 1111–1122, 2011.

[9] S. Ruggles, K. Genadek, R. Goeken, J. Grover, and M. Sobek, “Integratedpublic use microdata series: Version 6.0 [machine-readabledatabase].” Minneapolis: University of Minnesota, 2015.

[10] X. Han, J. Li, H. Gao, and C. Yang, “SEPT: an efficient skylinejoin algorithm on massive data,” *Knowledge and Information Systems*,vol. 43, no. 2, pp. 355–388, 2015.

[11] X. Han, J. Li, and H. Gao, “TDEP: efficiently processing top-k dominatingquery on massive data,” *Knowledge and Information Systems*,vol. 43, no. 3, pp. 689–718, 2015.

[12] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Talbot,K. Elmeleegy, and R. Sears, “Online aggregation and continuous querysupport in mapreduce,” in *Proceedings of the 2010 ACM SIGMODInternational Conference on Management of Data*, 2010, pp. 1115–1118.

[13] N. Pansare, V. R. Borkar, C. Jermaine, and T. Condie, “Online aggregationfor large mapreduce jobs,” *Proceedings of the Vldb Endowment*,vol. 4, no. 11, pp. 1135–1145, 2011.

[14] S. Chaudhuri, G. Das, and U. Srivastava, “Effective use of block-levelsampling in statistics estimation.” in *Proceedings of the 2004 ACMSIGMOD International Conference on Management of Data*, 2004, pp.287–298.

[15] A. Doucet, M. Briers, and S. Sncal, “Efficient block sampling strategiesfor sequential monte carlo methods,” *Journal of Computational andGraphical Statistics*, vol. 15, no. 3, pp. 693–711, 2012.

[16] A. Pol and C. Jermaine, “Relational confidence bounds are easy withthe bootstrap,” in *Proceedings of the 2005 ACM SIGMOD InternationalConference on Management of Data*, 2005, pp. 587–598.

**MapReduce**

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

What is MapReduce?

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

Generally MapReduce paradigm is based on sending the computer to where the data resides!

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

Map stage : The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

Reduce stage : This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.

The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

MapReduce Algorithm

Inputs and Outputs (Java Perspective)

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job: (Input) <k1, v1> -> map -> <k2, v2>-> reduce -> <k3, v3>(Output).

Input Output

Map <k1, v1> list (<k2, v2>)

Reduce <k2, list(v2)> list (<k3, v3>)

Terminology

PayLoad - Applications implement the Map and the Reduce functions, and form the core of the job.

Mapper - Mapper maps the input key/value pairs to a set of intermediate key/value pair.

NamedNode - Node that manages the Hadoop Distributed File System (HDFS).

DataNode - Node where data is presented in advance before any processing takes place.

MasterNode - Node where JobTracker runs and which accepts job requests from clients.

SlaveNode - Node where Map and Reduce program runs.

JobTracker - Schedules jobs and tracks the assign jobs to Task tracker.

Task Tracker - Tracks the task and reports status to JobTracker.

Job - A program is an execution of a Mapper and Reducer across a dataset.

Task - An execution of a Mapper or a Reducer on a slice of data.

Task Attempt - A particular instance of an attempt to execute a task on a SlaveNode.

Example Scenario

Given below is the data regarding the electrical consumption of an organization. It contains the monthly electrical consumption and the annual average for various years.

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Avg

1979 23 23 2 43 24 25 26 26 26 26 25 26 25

1980 26 27 28 28 28 30 31 31 31 30 30 30 29

1981 31 32 32 32 33 34 35 36 36 34 34 34 34

1984 39 38 39 39 39 41 42 43 40 39 38 38 40

1985 38 39 39 39 39 41 41 41 00 40 39 39 45

If the above data is given as input, we have to write applications to process it and produce results such as finding the year of maximum usage, year of minimum usage, and so on. This is a walkover for the programmers with finite number of records. They will simply write the logic to produce the required output, and pass the data to the application written.

But, think of the data representing the electrical consumption of all the largescale industries of a particular state, since its formation.

When we write applications to process such bulk data,

They will take a lot of time to execute.

There will be a heavy network traffic when we move data from source to network server and so on.

**Problem statement**

 the rapidly generated big data are not uploaded to a data center at once.

 Instead, the fresh big data is quickly stored in local servers temporarily.

 less efficient processing techniques and analytical methods for big data in a clustered environment

 Many new data are generated at high velocity.

**Screenshot**

