© 2015 IJSRSET | Volume 1 | Issue 6 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Heterogeneous Wireless Sensor Network for Big Data Analytics

Alpesh R. Sankaliya
Electronics & Communication Engineering Department, Government Polytechnic, Dahod, Gujarat, India

ABSTRACT

Wireless sensor networks (WSN) are collections of hypothetically large number of physical devices responsible for measuring environmental variables and of transmitting the data to one or more network(s). The data gathered through H-WSN (Heterogeneous Wireless Sensor Network) are to be dynamically process and is subjected to be analysed to monitor the considered issues and support the decision making. Data transmission is accomplished over radio links and routing is based on ad-hoc networking protocols. The sensor devices produce and collect large volume of data from physical-world where quality of data can also vary over time. The data can be represented as numerical measurement values or as symbolic descriptions of occurrence in the world. If we look on wireless sensor data (WSN) applications follows Militery applications, Environmental applications, Health applications, Home applications, Commercial applications and much more are varied dimension that needs big data analytics. All these analytics system need good performance and has to support uers's adaptively and also the quality, validity and trust of data collected by wireless sensors. These challenges requirement have attracted researchers to improve performance along with time and cost efficient.

Keywords: Wireless Sensor Network, Big Data Analytics, information technology, NOSQL, Hadoop distributed file system MapReduce, MPI, GPFS

I. INTRODUCTION

A sensor network is an infrastructure comprised of sensing (measuring), computing, and communication elements that gives an administrator the ability to instrument, observe, and react to events and phenomena in a specified environment. The administrator typically is a civil, governmental, commercial, or industrial entity. The environment can be the physical world, a biological system, or an information technology (IT) framework. Network(ed) sensor systems are seen by observers as an important technology that will experience major deployment in the next few years for a plethora of applications, not the least being national security Typical applications include, but are not limited to, data collection, monitoring, surveillance, and medical telemetry. In addition to sensing, one is often also interested in control and activation. Wireless sensor networks (WSNs) provide rapid , untethered access to information and computing, eliminating the barriers of distance, time, and location for many application sin national security, civilian search and rescue operations, surveillance, area/target monitoring, and many more.

The look at the big data not only tells that it is very large but also of varied dimensions and fast growing in data volume. As a result the suitable techniques and technologies are needed to collect the data from different sources and thereafter to organize, transform, manage and analyse efficiently the same for intended purpose. The analytical techniques may include data mining, cluster analysis etc. to extract and learn complex patterns that empowers decision makers to make intelligent decision and text analysis[1]. The technology domain that support to house and manage and utilize big data to facilitate analysis may include enterprise data warehousing, visually represent results, map reduce and Hadoop, NOSQL database. The requirements of different sectors needing infrastructure for Big data management and their analysis are different. It is experienced in diversified areas like Health care, Environment moitoring Astronomical data analysis etc. populate big data volume in short span of time increasing the volume exponentially. The data resource housed in big data repository has potential information to be analysed to make meaningful conclusions in case of monitoring as well as decision making.

The digital revolution and the fast increase of information exchange have generated interesting research issues, under the domain of big data analytics, as cyber security, intellectual property rights, digital evidence required in a cybercrime. The scientific research dry and it advances in the field of electric vehicals generate very large volume of logger data collected from a set of sensors fitted in vehicles[2]. The logger data are in the initial set store as flat file. These data are predominantly unstructured and non-relational but volume wise and nature wise it can safely regardless as big data. This big data set are very vital and needs appropriate analysis with high performance as well as computational efficiency, further the framework which is develop is expected to be scalable to accomadate fast growing data and ensuring its robust, security ever the communication channel in the network. This research needs Hadoop build architecture to manage distributed data over a network in a real time mode of data collection and its subsequent storage. The system should give support for high performance, parallel processing and efficient retrival for decision making.

CHALLENGES WITH WIRELESS SENSOR DATA

Sensor data brings numerious challenges with it in the context of data collection, storage and processing. This is because sensor data processing often requires efficient and real-time processing from massive and the nature of such data tends to be dynamic, heterogeneous, untidy and sometimes untrustworthy. This kind of large scale unstructured multi- dimensional data is known as Big Data[3]. Any analysis of such data will be cumbersome and complex for performing certain kind of analyses. Some of these challenges may be as follows:

- **Limited hardware:** Each node has limited processing, storage, and communication capabilities, and limited energy supply and bandwidth.
- **Limited support for networking:** The network is peer-to-peer, with a mesh topology and dynamic, mobile, and unreliable connectivity.
- **Limited support for software development:** The tasks are typically real-time and massively distributed, involve dynamic collaboration among nodes, and must handle multiple competing events.
- **Data collection** is a huge challenge in the context of sensor processing because of the natural errors and incompleteness in the collection process. Some sensors often have limited battery life, because of which many of the sensors in a network may not be able to collect or transmit their data over large periods of time. The errors in the underlying data may lead to uncertainty of the data representation. Therefore, methods need to be designed to process the data in the presence of uncertainty.
- Sensors are often designed for applications which require **real-time processing**. This requires the design of efficient methods for stream processing. Such algorithms need to be executed in one pass of the data, since it is typically not often possible to store the entire data set because of storage and other constraints.
- The large volumes of data lead to huge challenges in terms of **storage and processing of the data**. It has been estimated that since 2008, the number of internet-connected devices has exceeded the number of people on the planet. Thus, it is clear that the amount of machine generated data today greatly exceeds the amount of human generated data, and this gap is only likely to increase in the forseeable future. This is widely known as the *big data* problem in the context of analytical applications, or the *information overload problem* in stream processing.
- In many cases, it is critical to perform ***in-network processing***, wherein the data is processed within the network itself, rather than at a centralized service. This needs effective design of distributed processing algorithms, wherein queries and other mining algorithm can be processed within the network in real time.[4]

In spite of the associated complexities, the main goal is to comprehensively study this data without losing any important information. Also, the proliferation of data everyday poses a significant challenge towards accumulation, integration and storage of it. Such a rapid growth demands extensive scalability support from the computational systems. As the size of data increases, so does the query time for retrieving any information from it. Hence, it is important that the query method leverages the capabilities of the internal infrastructure to retrieve the required information from the datasets in minimum time. Hence, these factors have to be holistically considered while designing and implementing the data repository. Wireless sensor network is responsible for stpring, sharing, searching and analyzing data from heterogeneous devices [5]. So, collection data from different types of sensors make a focus on analyzing those data and make in knowledgeable data.

The work involves identifying and implementing complex algorithms in the field of Wireless sensor data research using MapReduce. Further, this dataset can be integrated with other entities such as weather data, traffic information, topology details, etc. This can further strengthen the use of Hadoop in harvesting user-customized information from heterogeneous datasets [6].

II. METHODS AND MATERIAL

A. Research Design And Methodology

Heterogeneous sensors data separates functionalities into a distributed decision system and a sensing system to overcome the limitations of a homogeneous system. The data are unique dataset that collected driving data during real time commute conditions. The main goal is to comprehensively study this data without losing any important information. Hadoop was chosen as the software framework for analyzing the collected heterogeneous sensors data. It is an open source tool. Hadoop distributed file system (HDFS) and MapReduce (MR) were the crucial components used in this data analytics. Experimental setup consisted of multiple Linux VMs configured over a cluster of physical servers. The raw data was purged anHadoop distributed file system (HDFS) and MapReduce (MR) were the based omponents used in this data analytics [7]. Appropriate map and reduce functions were programmed to retrieve information.

B. MapReduce

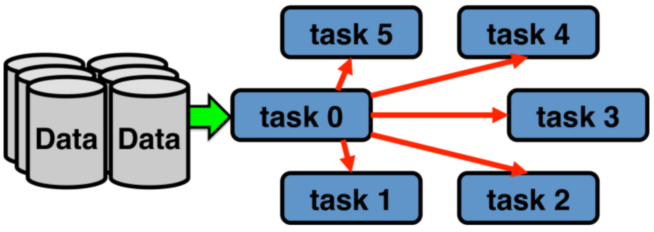
MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster.

In order to appreciate what map-reduce brings to the table; I think it is most meaningful to contrast it to what I call *traditional* computing problems. I define "traditional" computing problems as those which use libraries like MPI, OpenMP, CUDA, or pthreads to produce results by utilizing multiple CPUs to perform some sort of numerical calculation concurrently. Problems that are well suited to being solved with these traditional methods typically share two common features:

1. **They are cpu-bound**: the part of the problem that takes the most time is doing calculations involving floating point or integer arithmetic
2. **Input data is gigabyte-scale**: the data that is necessary to describe the conditions of the calculation are typically less than a hundred gigabytes, and very often only a few hundred megabytes at most

i. Traditional Parallel Applications

To illustrate these differences, the following schematic depicts how your typical traditionally parallel application works.



The input data is stored on some sort of remote storage device (a SAN, a file server serving files over NFS, a parallel Lustre or GPFS filesystem, etc; **grey cylinders**). The compute resources or elements (**blue boxes**) are abstract units that can represent MPI ranks, compute nodes, or threads on a shared-memory system.

Upon launching a traditionally parallel application,

- A master parallel worker (MPI rank, thread, etc) reads the input data from disk (**green arrow**).
- The master worker then divides up the input data into chunks and sends parts to each of the other workers (**red arrows**).
- All of the parallel workers compute their chunk of the input data
- All of the parallel workers communicate their results with each other, then continue the next iteration of the calculation

In some cases multiple workers may use a parallel I/O API like MPI-IO to collectively read input data, but the filesystem on which the input data resides must be a high-performance filesystem that can sustain the required device-and network-read bandwidth.

The fundamental limit to scalability here is step #1–the process of reading the input data (**green arrow**) is performed serially. Even if you can use MPI-IO to perform the data ingestion in parallel, the data is separated from the compute resources (blue squares) by some pipe through which data can flow at some finite rate. While it is possible to increase the speed of this connection between your data and your compute resources by throwing more money at it (e.g., buying fast SSDs, faster storage networking, and/or more parallel storage servers), the cost of doing this does not scale linearly.

Data-Intensive Applications

The map-reduce paradigm is a completely different way of solving a certain subset of parallelizable problems that gets around the bottleneck of ingesting input data from disk (that pesky green arrow). Whereas traditional parallelism brings *the data to the compute*, map-reduce does the opposite–it brings *the compute to the data*:

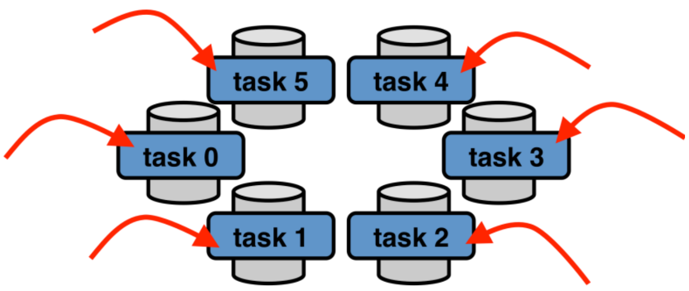


Figure 1 : Map Reduce Paradigm

In map-reduce, the input data is *not* stored on a separate, high-capacity storage system. Rather, the data exists in little pieces and is permanently stored on the compute elements. This allows our parallel procedure to follow these steps:

1. We don't have to move any data since it is pre-divided and already exists on nodes capable of acting as computing elements
2. All of the parallel worker functions are sent to the nodes where their respective pieces of the input data already exist and do their calculations
3. All of the parallel workers communicate their results with each other, move data if necessary, then continue the next step of the calculation

Thus, the only time data needs to be moved is when all of the parallel workers are communicating their results with each other in step #3. There is no more serial step where data is being loaded from a storage device before being distributed to the computing resources because the data already exists on the computing resources.

C. Hadoop - A Map-Reduce Implementation

Now that we've established a description of the map-reduce paradigm and the concept of bringing compute to the data, we are equipped to look at Hadoop, an actual implementation of map-reduce.

D. The Magic of HDFS (Hadoop Distributed File System)

The idea underpinning map-reduce–bringing compute to the data instead of the opposite–should sound like a very simple solution to the I/O bottleneck inherent in traditional parallelism. However, the devil is in the details, and implementing a framework where a single large file is transparently diced up and distributed across multiple physical computing elements (all while appearing to remain a single file to the user) is not trivial.

Hadoop, perhaps the most widely used map-reduce framework, accomplishes this feat using HDFS, the Hadoop Distributed File System. HDFS is fundamental to Hadoop because it provides the data chunking and distribution across compute elements necessary for map-reduce applications to be efficient. Since we're now talking about an actual map-reduce implementation and not an abstract concept; let's refer to the abstract *compute elements* now as *compute nodes*.

HDFS exists as a filesystem into which you can copy files to and from in a manner not unlike any other filesystem. Many of the typical commands for manipulating files (ls, mkdir, rm, mv, cp, cat,tail, and chmod, to name a few) behave as you might expect in any other standard filesystem (e.g., Linux's ext4).

The magical part of HDFS is what is going on just underneath the surface. Although it appears to be a filesystem that contains files like any other, in reality those files are distributed across multiple physical compute nodes:

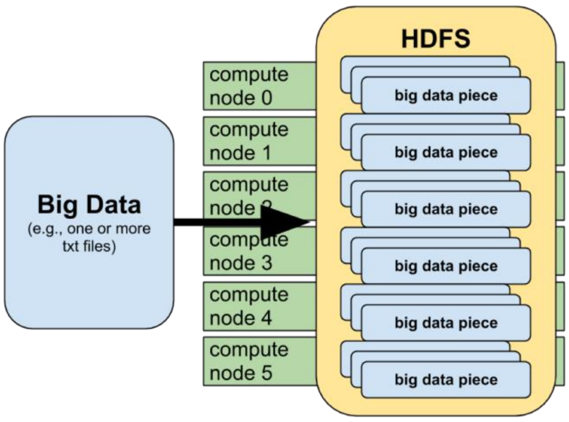


Figure 2 : HDFS

When you copy a file into HDFS as depicted above, that file is transparently sliced into 64 MB "chunks" and replicated three times for reliability. Each of these chunks are distributed to various compute nodes in the Hadoop cluster so that a given 64 MB chunk exists on three independent nodes. Although physically chunked up and distributed in triplicate, all of your interactions with the file on HDFS still make it appear as the same single file you copied into HDFS initially. Thus, HDFS handles the entire burden of slicing, distributing, and recombining your data for you.

E. Map-Reduce Jobs

HDFS is an interesting technology in that it provides data distribution, replication, and automatic recovery in a user-space filesystem that is relatively easy to configure and, conceptually, easy to understand. However, its true utility comes to light when map-reduce jobs are executed on data stored in HDFS.

As the name implies, map-reduce jobs are principally comprised of two steps: the map step and the reduce step. The overall workflow generally looks something like this:

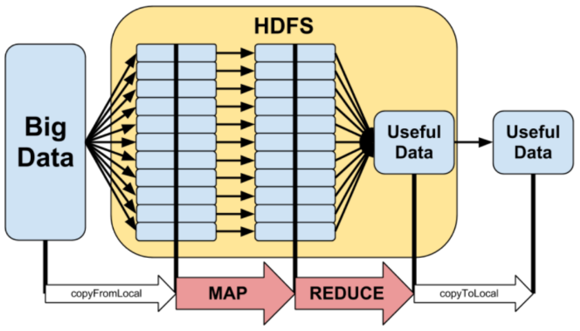


Figure 3 : Program Flow of a map-reduce Application

The left half of the diagram depicts the HDFS magic described in the previous section, where the hadoop dfs -copyFromLocal command is used to move a large data file into HDFS and it is automatically replicated and distributed across multiple physical compute nodes. While this step of moving data into HDFS is not strictly a part of a map-reduce job (i.e., your dataset may already have a permanent home on HDFS just like it would any other filesystem), a map-reduce job's input data must already exist on HDFS before the job can be started

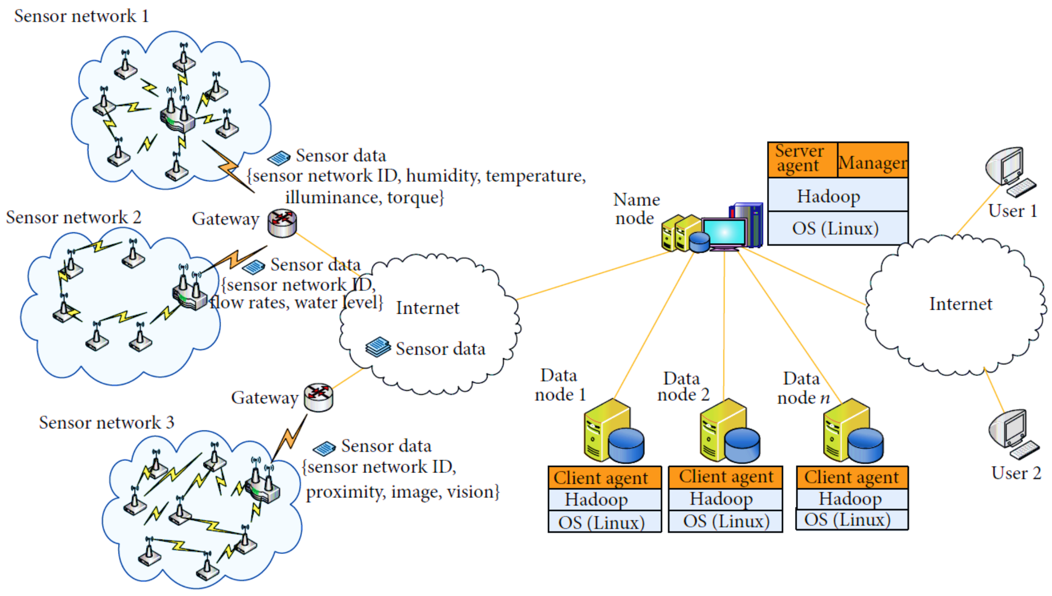


Figure 4 : Overall design of the Hadoop Framework

Figure 4 shows the overall structure of a Hadoop framework based on a wireless sensor network. In this figure, each sensor network group (sensor network 1, sensor network 2, and sensor network 3) has different sensors and purposes as well as generating a variety of sensing data. For example, sensor network 1 collects sensing data regarding environmental information (humidity, temperature, carbon dioxide, carbon monoxide, ozone, etc.). Sensor network 2 collects sensing data regarding historical information (water level, rainfall, etc.). Sensor network 3 collects sensing data regarding video information (proximity distance, image, vision, etc.). The collected sensing data will be very large in size; hence, it is called Big Data. These data are first stored in the local file system of the namenode. Stored data are again stored in the distributed file systems based on the Hadoop framework through MapReduce. The Hadoop framework is comprised of namenodes and datanodes. Each node has either a server agent or a client agent in order to monitor the Hadoop framework during MapReduce.

III. CONCLUSION & PROPOSED WORK

Scientific community has collected large amounts of raw data from experiments and simulation. This proliferation of data everyday poses a significant challenge towards its accumulation. Different data analytics methods of data analytics can be performed on wireless sensors datasets. In spite of all the involved complexities, the main goal of this study is to demonstrate the benefits of big data analytics.

The research targets are center around:

- Simplification of data formats for storage of data coming from heterogeneous sensors in real time mode.
- Real time information retrival and processing of filter and cluster data relevant to objective to open up new possibilities.
- Statistical analysis of modeling data to extract hidden patterns associations / relationships between selected parameters.
- Demostrate the benefits of heterogeneous data analysis.
- Enable user's capability to pre-process the data during the query as per their requirements.
- Monitor the performance of the sensor's activity to achieve the trustable and reliable data.
- Design Framework for heterogeneous big data analytics

IV. REFERENCES

[1] Romer Kay and Mattern F. (2004). The Design Space of Wireless Sensor Networks, IEEE Wireless Communications.

[2] Mhatre V and Rosenberg C. (2004). Homogeneous vs. Heterogeneous Clustered Sensor Networks : A Comparative Study. Proceedings of IEEE International Conference on Communications (ICC).

[3] Yuan L, and Gui C. (2004). Applications and Design of Heterogeneous and Broadband Advanced Sensor Networks (Basenets).

[4] Big-Data in the Cloud: Converging Technologies, Intel IT centre, September 2014.

[5] Wireless Sensor Networks: Technology, Protocols, and Applications, by Kazem Sohraby, Daniel Minoli, and Taieb Znati

[6] "Intelligent services for Big Data science," C. Dobrea, F. Xhafab, University Politehnica of Bucharest, Romana, Elsevier, 9 August 2013.

[7] "A study of electrical vehicle Data Analytics," Vamshi K. Bolly, John A . Springer, J. Eric Dietz!, Computer and Information Technology, College of Technology, Purdue University.

[8] "Analytics over large-scale multidimensional data: the Big Data revolution," A. Cuzzocrea, I. Y. Song, and K. C. Davis. in Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP, ACM, October, 2011.

[9] "Managing and Mining sensor data" ,Charu C. Aggrawal, IBM T. J. Watson Research Center Yorktown Heights, NY 10598

[10] "An introduction to sensor data analytics" ,Charu C. Aggrawal, IBM T. J. Watson Research Center Yorktown Heights, NY 10598

[11] "Apache Hadoop", The Apache Software Foundation, http://hadoop.apache.org, February, 2014.

[12] Harnessing Hadoop: Understanding the Big Data Processing Options for Optimizing Analytical Workloads, www.cognizant.com, cognizant 20-20 Insights

[13] M. Caccamo, L.Y. Zhang, L. Sha, G. Buttazzo, An implicit prioritized access protocol for wireless sensor networks, in: Proc. IEEE Real-Time Systems Symp., December 2002, pp. 39–48.