

e-PGPathshala

Subject : Computer Science

Paper: Data Analytics

Module: Evolution of Analytical Scalability

Module No: CS/DA/3

Quadrant 1 – e-text

1.1 Introduction

The amount of data organizations process continues to increase. More data cross the internet every second than were stored in the entire internet just 20 years ago. This gives companies an opportunity to work with many petabytes of data in a single data set—and not just from the internet. For instance, it is estimated that Walmart collects more than 2.5 petabytes of data every hour from its customer transactions. A petabyte is one quadrillion bytes, or the equivalent of about 20 million filing cabinets' worth of text. An exabyte is 1,000 times that amount, or one billion gigabytes. The old methods for handling such huge data won't work anymore.

Hence we need technologies that might handle and tame the big data. Few of such kind are as show in figure 1, but there are many more:

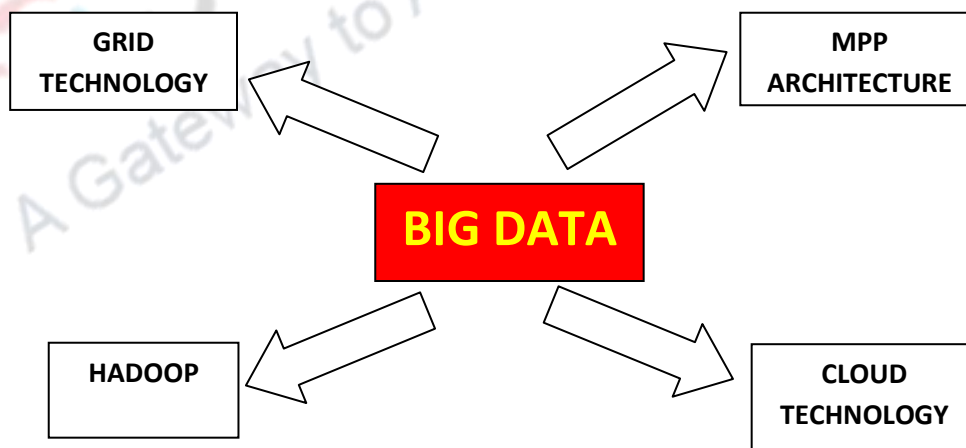


Figure 1. Big data technologies

We shall discuss about the above technologies in detail as we proceed further in this module. But before that the readers must be aware of how analytics and data environment converged together for making big data analytics more effective.

1.2 Learning Objectives

- To understand the evolution of analytical scalability
- To know about technologies that can handle big data

1.3 The Convergence of the Analytic and Data Environment

1.3.1 Traditional Analytic Architecture

Traditional analytics collects data from heterogeneous data sources and we had to pull all data together into a separate analytics environment to do analysis which can be an analytical server or a personal computer with more computing capability. The heavy processing occurs in the analytic environment as shown in figure 2.

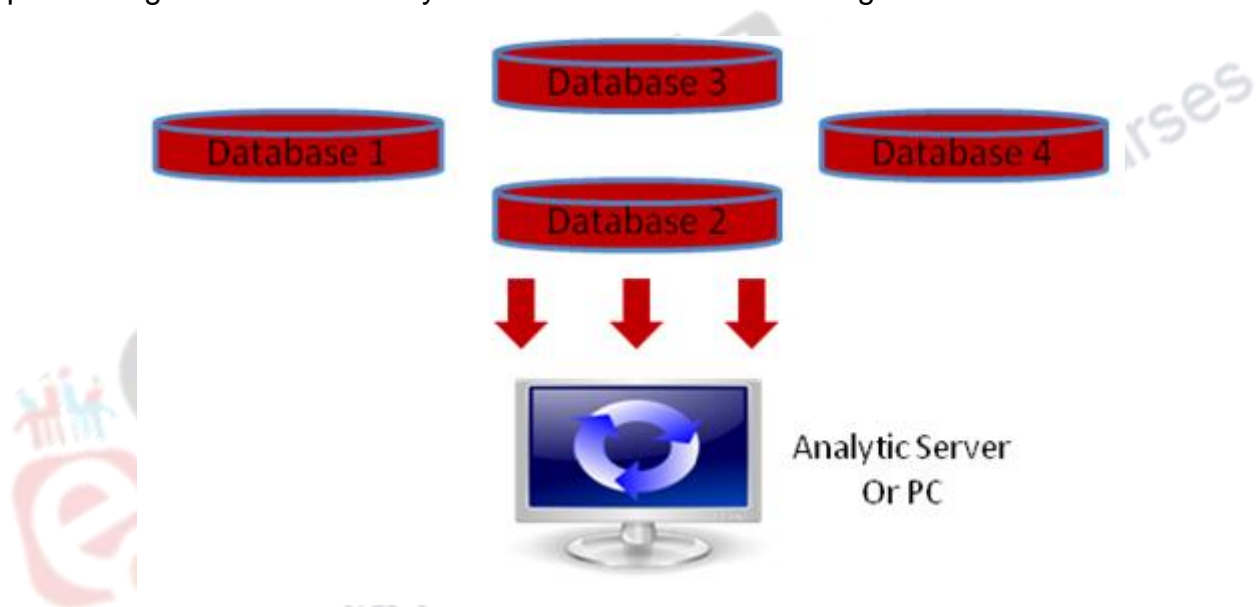


Figure 2. Traditional Analytic Architecture

In such environments, shipping of data becomes a must, which might result in issues related with security of data and its confidentiality.

1.3.2 Modern In-Database Architecture

Data from heterogeneous sources are collected, transformed and loaded into data warehouse for final analysis by decision makers. The processing stays in the database where the data has been consolidated. The data is presented in aggregated form for querying. Queries from users are submitted to OLAP (online analytical processing) engines for execution. Such in-database architectures are tested for their query throughput rather than transaction throughput as in traditional database environments. More of metadata is required for directing the queries which helps in reducing the time taken for answering queries and hence increase the query throughput. Moreover the

data in consolidated form are free from anomalies, since they are preprocessed before loading into warehouses which may be used directly for analysis. The modern in-database architecture is shown in figure 3.

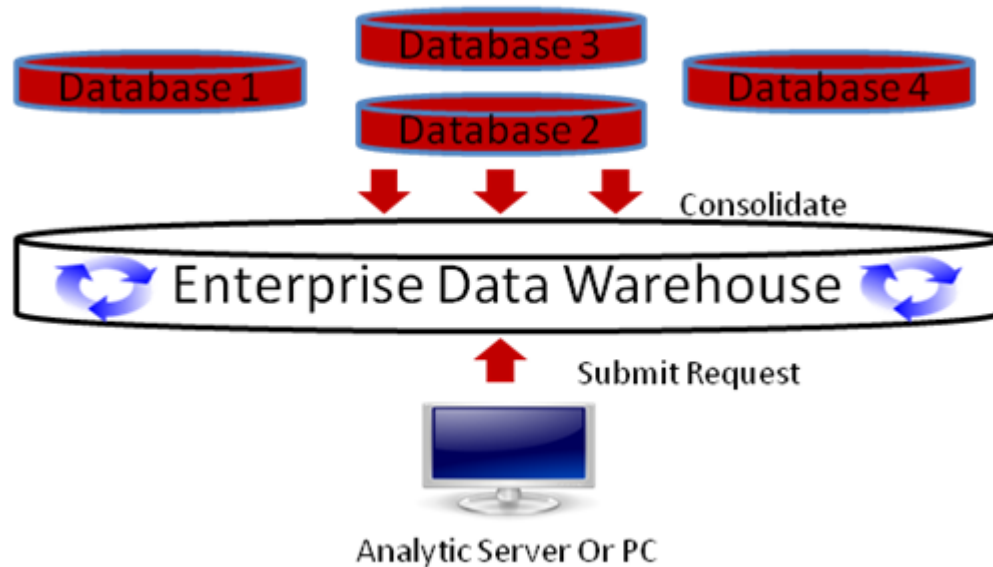


Figure 3. In-Database Architecture

1.3.3 Massively Parallel Processing (MPP)

Massive Parallel Processing (MPP) is the “shared nothing” approach of parallel computing. It is a type of computing wherein the process is being done by many CPUs working in parallel to execute a single program.

One of the most significant differences between a Symmetric Multi-Processing or SMP and Massive Parallel Processing is that with MPP, each of the many CPUs has its own memory to assist it in preventing a possible hold up that the user may experience with using SMP when all of the CPUs attempt to access the memory simultaneously. The MPP architecture is given in figure 4.

The salient feature of MPP systems are:

- Loosely coupled nodes
- Nodes linked together by a high speed connection
- Each node has its own memory

- Disks are not shared, each being attached to only one node – shared nothing architectures

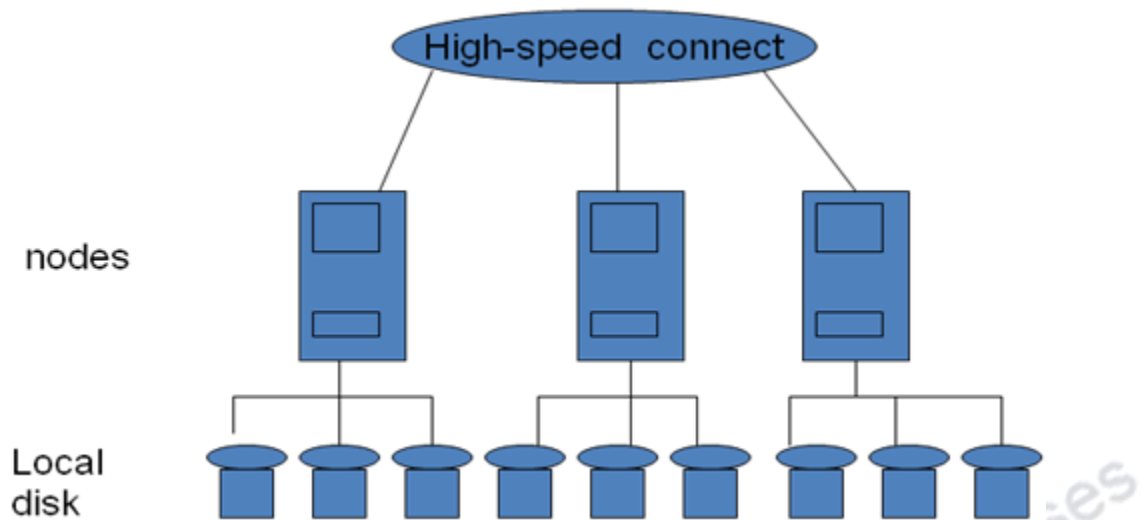


Figure 4. MPP Architecture

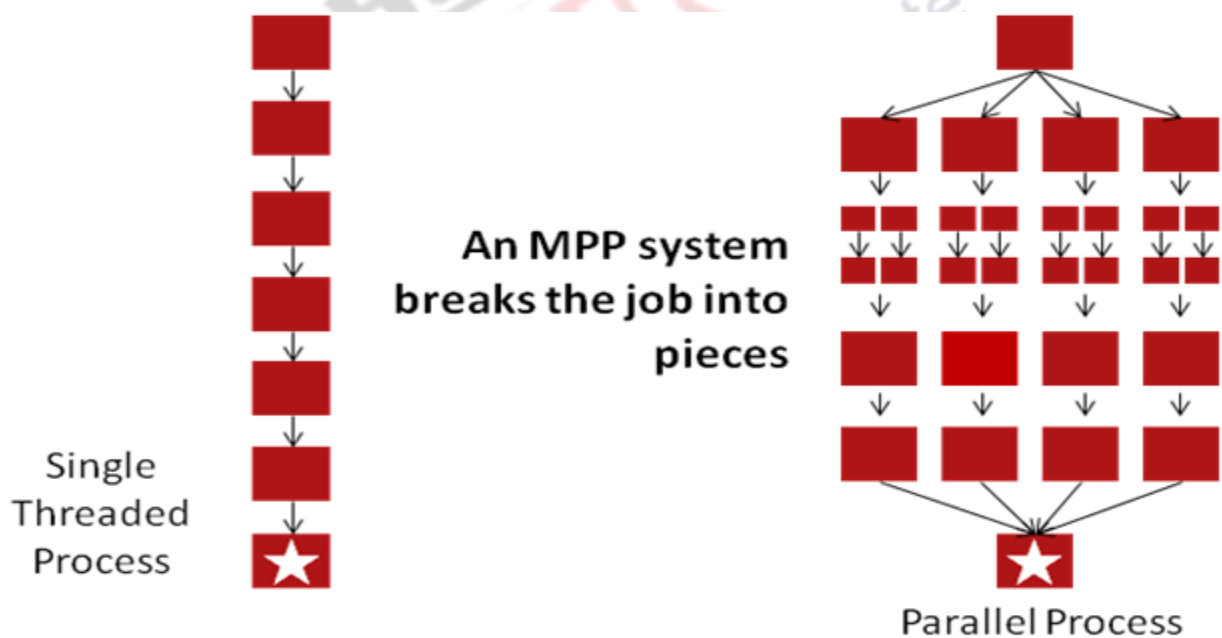


Figure 5. Execution of tasks in MPP

The idea behind MPP is really just that of the general parallel computing (figure 5) wherein the simultaneous execution of some combination of multiple instances of programmed instructions and data on multiple processors in such a way that the result can be obtained more effectively.

MPP uses shared distributed lock manager to maintain the integrity of the distributed resources across the system. CPU power that can be made available in a MPP is dependent upon number of nodes that can be connected. MPP systems build in redundancy to make recovery easy. MPP systems have resource management tools to manage the CPU and disk space and Query optimization.

1.3.4 The Cloud Computing

Cloud computing is the delivery of computing services over the Internet. Cloud services allow individuals and businesses to use software and hardware that are managed by third parties at remote locations. Examples of cloud services include online file storage, social networking sites, webmail, and online business applications. The cloud computing model allows access to information and computer resources from anywhere that a network connection is available. Cloud computing provides a shared pool of resources, including data storage space, networks, computer processing power, and specialized corporate and user applications.

McKinsey and Company in the paper 'Clearing the Air on Cloud Computing' (March 2009) has indicated the following as characteristic features of cloud:

- 1) Mask the underlying infrastructure from the user
- 2) Be elastic to scale on demand
- 3) On a pay-per-use basis
- 4) National Institute of Standards and Technology (NIST)
- 5) On-demand self-service
- 6) Broad network access
- 7) Resource pooling
- 8) Rapid elasticity
- 9) Measured service

There are two types of cloud environment:

1. Public Cloud

- The services and infrastructure are provided off-site over the internet
- Greatest level of efficiency in shared resources
- Less secured and more vulnerable than private clouds


2. Private Cloud

- Infrastructure operated solely for a single organization
- The same features of a public cloud

- Offer the greatest level of security and control
- Necessary to purchase and own the entire cloud infrastructure

1.3.5 Grid Computing

Grid computing is a form of distributed computing whereby a "super and virtual computer" is composed of a cluster of networked, loosely coupled computers, acting in concert to perform very large tasks. Grid computing (Foster and Kesselman, 1999) is a growing technology that facilitates the executions of large-scale resource intensive applications on geographically distributed computing resources. Facilitates flexible, secure, coordinated large scale resource sharing among dynamic collections of individuals, institutions, and resource

	<p>Do you Know?</p>
<p>SETI@Home (Source: https://en.wikipedia.org/wiki/SETI@home)</p>	
<p>SETI@home ("SETI at home") is an Internet-based public volunteer computing project employing the BOINC software platform, hosted by the Space Sciences Laboratory, at the University of California, Berkeley, in the United States. SETI is an acronym for the Search for Extra-Terrestrial Intelligence. Its purpose is to analyze radio signals, searching for signs of extra terrestrial intelligence, and is one of many activities undertaken as part of SETI.</p> <p>SETI@home is the second large-scale use of distributed computing over the Internet for research purposes. Along with MilkyWay@home and Einstein@home, it is the third major computing project of this type that has the investigation of phenomena in interstellar space as its primary purpose.</p>	
<p>GARUDA (Source: https://en.wikipedia.org/wiki/GARUDA)</p>	
<p>GARUDA (Global Access to Resource Using Distributed Architecture) is India's Grid Computing initiative connecting 17 cities across the country. GARUDA is a collaboration of science researchers and experimenters on a nationwide grid of computational nodes, mass storage and scientific instruments that aims to provide</p>	

the technological advances required to enable data and compute intensive science for the 21st century.

The GARUDA High-Speed network is a Layer 2/3 MPLS Virtual Private Network (VPN) connecting select 45 institutions across 17 cities at 10/100 Mbit/s with Stringent Service Level Agreements with the service provider. This Grid is a precursor to the Gigabit speed nationwide Wide Area Network (WAN) connecting high performance computing resources and scientific instruments for seamless collaborative research and experiments.

GARUDA has adopted a pragmatic approach for using existing Grid infrastructure and Web Services technologies. The deployment of grid tools and services for GARUDA will be based on a judicious mix of in-house developed components, the Globus Toolkit (GT), industry grade & open source components. The Foundation phase GARUDA will be based on stable version of GT4. The resource management and scheduling in GARUDA is based on a deployment of industry grade schedulers in a hierarchical architecture. At the cluster level, scheduling is achieved through Load Leveler for AIX platforms and Torque for Linux clusters.

The GARUDA portal which provides the user interface to the Grid resources hides the complexity of the Grid from the users. It allows submission of both sequential and parallel jobs and also provides job accounting facilities. Problem Solving Environment (PSE) in the domains of Bio-informatics, and Community Atmospheric Model support the entire cycle of problem solving for the specific domains by supporting problem formulation, algorithm selection, numerical simulation and solution visualization.

1.3.5.1 Comparison of grids and conventional supercomputers

“Distributed” or “grid” computing in general is a special type of parallel computing that relies on complete computers (with onboard CPUs, storage, power supplies, network interfaces, etc.) connected to a network (private, public or the Internet) by a conventional network interface producing commodity hardware, compared to the lower efficiency of designing and constructing a small number of custom supercomputers. The primary performance disadvantage is that the various processors and local storage areas do not have high-speed connections. This arrangement is thus well-suited to applications in which multiple parallel computations can take place independently, without the need to communicate intermediate results between processors. The high-end scalability of geographically dispersed grids is generally favorable, due to the low need for connectivity between nodes relative to the capacity of the public Internet.

There are also some differences in programming and deployment. It can be costly and difficult to write programs that can run in the environment of a supercomputer, which may have a custom operating system, or require the program to address concurrency issues. If a problem can be adequately parallelized, a “thin” layer of “grid” infrastructure can allow conventional, standalone programs, given a different part of the same problem, to run on multiple machines. This makes it possible to write and debug on a single conventional machine, and eliminates complications due to multiple instances of the same program running in the same shared memory and storage space at the same time.

1.3.6 Hadoop

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. Two main building blocks inside this runtime environment are MapReduce and Hadoop Distributed File System (HDFS). Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the HDFS are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

Minimally, applications specify the input/output locations and supply map and reduce functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters, comprise the job configuration. The Hadoop job client then submits the job (jar/executable etc.) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

1.3.6.1 Working of MapReduce

Let's assume there are 20 terabytes of data and 20 MapReduce server nodes for a project. The steps are summarized as follows as diagrammatically represented as in figure 5:

- Distribute a terabyte to each of the 20 nodes using a simple file copy process
- Submit two programs(Map, Reduce) to the scheduler
- The map program *finds the data* on disk and *executes* the logic it contains
- The results of the map step are then passed to the reduce process to *summarize* and *aggregate* the final answers

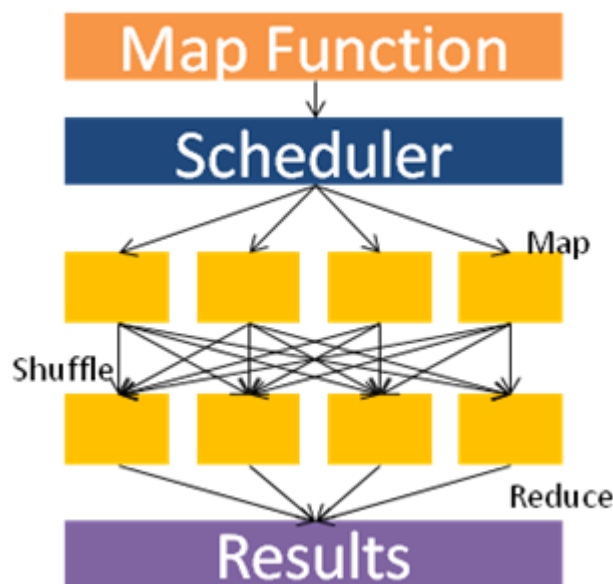


Figure 5. Working of MapReduce

In general MapReduce is Good for the following:

- Lots of input, intermediate, and output data
- Batch oriented datasets (ETL: Extract, Load, Transform)
- Cheap to get up and running because of running on commodity hardware

However, MapReduce is not suitable for the following:

- Fast response time
- Large amounts of shared data

- c) CPU intensive operations (as opposed to data intensive)
- d) NOT a database
 - i. No built-in security
 - ii. No indexing, No query or process optimizer
 - iii. No knowledge of other data that exists

Finally in real world we find few technologies that can integrate and work together for aiding big data analytics:

- 1) Databases running in the cloud, e.g., SimpleDB, Microsoft SQL Azure
- 2) Databases including MapReduce functionality, e.g., Teradata Aster, Oracle
- 3) MapReduce can be run against data sourced from a database, e.g., CouchDB, mongoDB
- 4) MapReduce can also run against data in the cloud, e.g., cloudmapreduce

Summary

- Analytics has evolved along with technology.
- Currently many technologies are available to handle big data efficiently.



Case Studies

A) Streamlining Healthcare Connectivity with Big Data

Source:

http://hadoopilluminated.com/hadoop_illuminated/cached_reports/Cloudera_Case_Study_Healthcare.pdf

Problem: A health IT company instituted a policy of saving seven years of historical claims and remit data, but its in-house database systems had trouble meeting the data retention requirement while processing millions of claims every day.

Solution: A Hadoop system allows archiving seven years' claims and remit data, which requires complex processing to get into a normalized format, logging terabytes of data

generated from transactional systems daily, and storing them in CDH for analytical purposes.

Cluster/Data size: 10+ nodes pilot; 1TB of data / day

Initially the organization downloaded Hadoop from Apache and configured it to run on 10 Dell workstations that were already in house. Once the small Hadoop cluster showed its functionality and demonstrated value, the team decided to make a commitment to the platform, but would need support to do so. When evaluating various Hadoop distributions and management vendors, they recognized that Cloudera was different: its Hadoop distribution — CDH — is 100% Apache open source. This allows Cloudera customers to benefit from rapid innovations in the open source community while also taking advantage of enterprise-grade support and management tools offered with the Cloudera Enterprise subscription.

B) NetApp

Source:

http://hadoopilluminated.com/hadoop_illuminated/cached_reports/Cloudera_Case_Study_NetApp.pdf

NetApp collects diagnostic data from its storage systems deployed at customer sites. This data is used to analyze the health of NetApp systems.

Problem: NetApp collects over 600,000 data transactions weekly, consisting of unstructured logs and system diagnostic information. Traditional data storage systems proved inadequate to capture and process this data.

Solution: A Cloudera Hadoop system captures the data and allows parallel processing of data.

Cluster/Data size: 30+ nodes; 7TB of data / month

AutoSupport collects over 600,000 data transactions weekly, consisting of unstructured logs and system diagnostic information. Approximately 40% of that data is transmitted during an 18-hour period each weekend, creating the potential for I/O bottlenecks that could affect service-level agreement (SLA) windows. AutoSupport data is growing at approximately 7 TB per month. Related storage requirements are doubling every 16 months. NetApp's AutoSupport team proactively identified the need to upgrade its storage environment in order to accommodate continued growth. A Cloudera Hadoop system was used to capture the data and allows parallel processing of data.