How to Plan Capacity for Hadoop Cluster

Deepak Mane

GCP – IT Performance Management

Tata Research Development and Design Center

Hadapsar, Pune INDIA

ABSTRACT

Apache Hadoop is an open-source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license. It supports the running of applications on large clusters of commodity hardware. We need an efficient , correct approach to build a large hadoop cluster with a large set of data having accuracy , speed . Capacity planning plays important role to decide choosing right hardware configuration for hadoop components . This paper describe sizing or capacity planning consideration for hadoop cluster and its components

1. Scope of Planning

The first rule to observe when planning like this is to know that there is really no one size fits all capacity planning. An attempt at that will only end in a customized disaster. The better idea is to create an individually managed plan for each service you plan to have in your IT infrastructure. There are, in fact, hardware requirements for everything from planning network infrastructure, messaging systems, collaboration systems, database systems, web servers, and file systems to building the server rooms or data center. There is a general rule of thumb that, if you have some monster hardware, your problems is over. Well, while that sounds pretty much a good logic, it doesn't mean your company can afford it. Following a simple basic rule of capacity planning, we can always be on top of the situation.

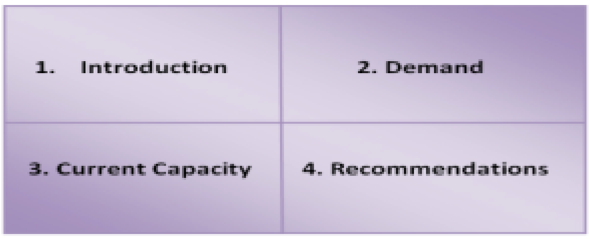
[raw][no_access]

[/no_access]

[ym_user_is package="1″]

[/ym_user_is][/raw]

[private]



This diagram (left) represents a typical template. It represents the thought process. First, to know what is on ground, and to determine current demand, and map that to current capacity, and the difference is in the lapses that a recommendation fixes.

There will be scenarios where this can be a startup project, where there exists no previous demand and no current capacity, but a recommendation needs to be made. In this scenario, the bottom line has to be made.

2. Startup Company Hardware Capacity Plan for Hadoop Cluster

To understand how to make a hardware capacity plan for a startup, the following has to be known.

- Size and Budget—This implies knowing the human resource size of the company that will consume the services the hardware will provide and the location(s) in terms of: Are they housed in one building or are there other locations that will be a part of the setup? This will determine the type and quality of hardware to purchase. Most hardware has certain capacity limits. The likelihood is that the startup already falls within the acceptable limit of a particular hardware type. I have, however, seen a scenario where an active directory environment got saturated with less than 20 users. The mistake the person made? Running domain services on a single Pentium 4 CPU with 2GB of RAM. Nobody does that, not even the guys in Hollywood. Hardware comes with low-end and high-end capacity. Be sure to obtain the right kind for your environment based on your budgetary allocations. Identify the trade-offs and ensure to take deeper look at all the options.
- Business Services Requirements—This refers to the business applications that will run in the startup company. These include but are not limited to domain services, network services, and application infrastructure services (those that require disk reads and writes in heavy volumes in databases). Identifying all these business applications will not be a one-time event, because other needs will emerge that seemed unnecessary in the past only to become an urgency in the present and future. Knowing these will help shape the technical needs.
- Technical Services Requirements—This refers to all the nitty-gritty details on a per-application basis. For example, domain services require that all network interface cards (NICs), also known as network cards, function at a maximum capacity of 100mbps/gbps, and that all cable runs are the required category (CAT 5e, CAT 6, etc.) This may necessitate that all routers and switches function at specific levels (Level 3, Level 4). Other considerations will be disk revolution cycles (10,000RPM, etc); it will also decide if disk arrays will be used (IDE, SCSI, RAID 0-5, SSD, etc), or newer

technology. With performance considerations, physical or virtual load balancing, failover, clusters, and such technology have to be considered, along with their supporting hardware. Technically, some things are harder to add later, so it always makes some sense to have performing hardware at the beginning, typically processor and memory (DDRs). The location hardware considerations are not excluded: cooling systems, racks, power systems, and backups, among others.

- Utilization and Optimization Plan—Utilization planning involves using calculative mechanisms to determine storage consumption over time and to determine storage capacity. This is done by estimating the average size of documents. This is where the usage of across-the-board templates for content management comes into play. By using templates for all document types it is easier to determine the average size of file creation. The backup strategy of critical business data and related contents will help estimate the growth rate for facilities used for onsite and offsite storage.
- Assumptions and Constrains – This is one area that can ruin a hardware capacity plan. It is common in nearly every project. It is always assumed that everything will go well as planned. It is assumed that vendors will deliver on the dates they promised. It is assumed that funds will be available. It is assumed that the personnel will be available to implement. It is assumed that business processes and services will run smoothly while the implementation is ongoing. Truth be told, this is not always the case. A constraint, on the other hand, is the fact that all the assumptions are not going to hold true, and will therefore affect this plan

3. Growing Concern Hardware Capacity Plan

A growing concern is a company with existing infrastructure, which need to continually review the current hardware state and reappraise it in the light of new demands on the existing capacity, and make recommendations for repair, replace, or upgrade. This is especially necessary if the previous design was not future-proofed, where that means allowing for future possibilities of anything changing, either good or bad. For the greater part, a growing concern will not embark on a complete overhaul, but on systematic replacement based on the diverse needs of old services and new services. It is important to also state that even future-proofed designs still need to be revisited.

By using historical data collected over the lifespan of the existing infrastructure, it is easy to make an estimate for future usage if trends continue. This is also where governance and archiving methodologies come to the fore. A typical outcome of this situation can be documented in a way similar to this:

4. Hardware Selection

When planning an Hadoop cluster, picking the right hardware is critical. No one likes the idea of buying 10, 50, or 500 machines just to find out she needs more RAM or disk. Hadoop is not unlike traditional data storage or processing systems in that the proper ratio of CPU to memory to disk is heavily influenced by the workload. There are, of course, some guidelines and a reasonable base configuration, but some knowledge of the intended workload will greatly increase the likelihood of optimum utilization of the hardware

Hadoop hardware comes in two distinct classes: masters and workers. Master nodes are typically more robust to hardware failure and run critical cluster services. Loss of a master almost certainly means some kind of service disruption. On the other hand, worker nodes are expected to fail regularly. This directly impacts the type of hardware as well as the amount of money spent on these two classes of hardware. It is common that administrators, in an effort to reduce the proliferation of hardware profiles in the data center, will select a single hardware profile for all masters and a single profile for all workers. Those with deep pockets may find it even easier to purchase a single Hadoop node profile and simply ignore wasted disk on the masters,

A. Master Hardware Selection

For master nodes—the machines that run one or more of the NameNode, jobtracker, and secondary NameNode—redundancy is the name of the game. Each of these machines serves a critical function the cluster can't live without While proponents of Hadoop beat the commodity hardware drum, this is the place where people spend more money and spring for the higher-end features. Dual power supplies, bonded network interface cards (NICs), and sometimes even RAID 10 in the case of the NameNode storage device, are not uncommon to find in the wild. In general, master processes tend to be RAM-hungry but low on disk space consumption. The NameNode and jobtracker are also rather adept at producing logs on an active cluster, so plenty of space should be reserved on the disk or partition on which logs will be stored.

The operating system device for master nodes should be highly available. This usually means RAID-1 (a mirrored pair of disks). Since the OS does not consume a significant amount of space, RAID-10 or RAID-5 would be overkill and lead to unusable capacity. Most of the real work is done on the data devices, while the OS device usually only has to contend with logfiles in */var/log*.

Small clusters—clusters with fewer than 20 worker nodes—do not require much for master nodes in terms of hardware. A solid baseline hardware profile for a cluster of this size is a dual quad-core 2.6 Ghz CPU, 24 GB of DDR3 RAM, dual 1 Gb Ethernet NICs, a SAS drive controller, and at least two SATA II drives in a JBOD configuration in addition to the host OS device. Clusters of up to 300 nodes fall into the mid-size category and usually benefit from an additional 24 GB of RAM for a total of 48 GB. Master nodes in large clusters should have a total of 96 GB of RAM. Remember that these are baseline numbers meant to give you a place from which to start

B. NameNode considerations

The NameNode is absolutely critical to a Hadoop cluster and usually receives special treatment. There are three things a healthy NameNode absolutely requires in order to function properly: RAM, modest but dedicated disk, and to be left alone! As we covered previously, the NameNode serves all of its metadata directly from RAM. This has the obvious implication that all metadata must fit in physical memory. The exact amount of RAM required depends on how much metadata there is to maintain. Remember that the metadata contains the filename, permissions, owner and group data, list of blocks that make up each file, and current known location of each replica of each block. As you'd expect, this adds up.

There are subtleties to the NameNode metadata that you might not otherwise think much about. One instance of this is that the length of filenames actually starts to matter at scale; the longer the filename, the more bytes it occupies in memory. More dubious, though, is the small files problem. Each file is made up of one or more blocks and has associated metadata. The more files the NameNode needs to track, the more metadata it maintains, and the more memory it requires as a result. As a base rule of thumb, the NameNode consumes roughly 1 GB for every 1 million blocks. Again, this is a guideline and can easily be invalidated by the extremes.

NameNode disk requirements are modest in terms of storage. Since all metadata must fit in memory, by definition, it can't take roughly more than that on disk. Either way, the amount of disk this really requires is minimal—less than 1 TB.

While NameNode space requirements are minimal, reliability is paramount. When provisioning, there are two options for NameNode device management: use the NameNode's ability to write data to multiple JBOD devices, or write to a RAID device. No matter what, a copy of the data should always be written to an NFS (or similar) mounted volume in addition to whichever local disk configuration is selected. This NFS mount is the final hope for recovery when the local disks catch fire or when some equally unappealing, apocalyptic event occurs.[9] The storage configuration selected for production usage is usually dictated by the decision to purchase homogeneous hardware versus specially configured machines to support the master daemons. There's no single correct answer and as mentioned earlier, what works for you depends on a great many factors.

C. Secondary NameNode hardware

The secondary NameNode is almost always identical to the NameNode. Not only does it require the same amount of RAM and disk, but when absolutely everything goes wrong, it winds up being the replacement hardware for the NameNode. Future versions of Hadoop (which should be available by the time you read this) will support a highly available NameNode (HA NN) which will use a pair of the time you read this) will support a highly available NameNode (HA NN) which will use a pair of identical machines. When running a cluster with an HA NameNode, the standby or inactive NameNode instance performs the checkpoint work the secondary NameNode normally does.

D. Jobtracker hardware

Similar to the NameNode and secondary NameNode, the jobtracker is also memory-hungry, although for a different reason. In order to provide job and task level-status, counters, and progress quickly, the jobtracker keeps metadata information about the last 100 (by default) jobs executed on the cluster in RAM. This, of course, can build up very quickly and for jobs with many tasks, can cause the jobtracker's JVM heap to balloon in size. There are parameters that allow an administrator to control what information is retained in memory and for how long, but it's a trade-off; job details that are purged from the jobtracker's memory no longer appear in its web UI.

Due to the way job data is retained in memory, JobTracker memory requirements can grow independent of cluster size. Small clusters that handle many jobs, or jobs with many tasks, may require more RAM than expected. Unlike the NameNode, this isn't as easy to predict because the variation in the number of tasks from job to job can be much greater than the metadata in the NameNode, from file to file.

E. Worker Hardware Selection

When sizing worker machines for Hadoop, there are a few points to consider. Given that each worker node in a cluster is responsible for both storage and computation, we need to ensure not only that there is enough storage capacity, but also that we have the CPU and memory to process that data. One of the core tenets of Hadoop is to enable access to all data, so it doesn't make much sense to provision machines in such a way that prohibits processing. On the other hand, it's important to consider the type of applications the cluster is designed to support. It's easy to imagine use cases where the cluster's primary function is long-term storage of extremely large datasets with infrequent processing. In these cases, an administrator may choose to deviate from the balanced CPU to memory to disk configuration to optimize for storage-dense configurations.

Scenario 1) – For storage

Consider the case where a system ingests new data at a rate of 1 TB per day. We know Hadoop will replicate this data three times by default, which means the hardware needs to accommodate 3 TB of new data every day! Each machine also needs additional disk capacity to store temporary data during processing with MapReduce. A ballpark estimate is that 20-30% of the machine's raw disk capacity needs to be reserved for temporary data. If we had machines with 12 × 2 TB disks, that leaves only 18 TB of space to store HDFS data, or six days' worth of data.

Scenario 2) – For CPU

although in this case, the focus is how much a machine can do in parallel rather than how much data it can store. Let's take a hypothetical case where an hourly data processing job is responsible for processing data that has been ingested. If this job were to process 1/24th of the aforementioned 1 TB of data, each execution of the job would need to process around 42 GB of data. Commonly, data doesn't arrive with such an even distribution throughout the day, so there must be enough capacity to be able to handle times of the day when more data is generated. This also addresses only a single job whereas production clusters generally support many concurrent jobs.

Typically, each task needs between 2 GB and 4 GB of memory, depending on the task being performed. A machine with 48 GB of memory, some of which we need to reserve for the host OS and the Hadoop daemons themselves, will support between 10 and 20 tasks. Of course, each task needs CPU time. Now there is the question of how much CPU each task requires versus the amount of RAM it consumes. Worse, we haven't yet considered the disk or network I/O required to execute each task. Balancing the resource consumption of tasks is one of the most difficult tasks of a cluster administrator. Later, we'll explore the various configuration parameters available to control resource consumption between jobs and tasks.

with the goal of stuffing as much CPU, RAM, and disk into a single machine as possible. At some point, we collectively realized this was difficult and expensive. For many data center services, we moved to running “pizza boxes” and building in the notion of failure as a first-class concept. A few years later, we were confronted with another problem: many machines in the data center were drastically underutilized and the sheer number of machines was difficult to manage. This was the dawn of the great virtualization rush. Machines were consolidated onto a smaller number of beefy boxes, reducing power and improving utilization. Local disk was eschewed in favor of large storage area networks (SANs) and network attached storage (NAS) because virtual machines could now run on any physical machine in the data center. Now along comes Hadoop and everything you read says commodity, scale-out, share-nothing hardware, but what about the existing investment in blades, shared storage systems, and virtualized infrastructure?

Hadoop, generally speaking, does not benefit from virtualization. Some of the reasons concern the techniques used in modern virtualization, while others have more to do with the common practices that exist in virtualized environments. Virtualization works by running a hypervisor either in a host OS or directly on bare metal, replacing the host OS entirely. Virtual machines (VMs) can then be deployed within a hypervisor and have access to whatever hardware resources are allocated to them by the hypervisor. Historically, virtualization has hurt I/O performance-sensitive applications such as Hadoop rather significantly because guest OSes are unaware of one another as they perform I/O scheduling operations and, as a result, can cause excessive drive seek operations. Many virtualization vendors are aware of this and are working toward more intelligent hypervisors, but ultimately, it’s still slower than being directly on bare metal. For all the reasons you would not run a high-performance relational database in a VM, you should not run Hadoop in a VM.

Those new to Hadoop from the high-performance computing (HPC) space may look to use blade systems in their clusters. It is true that the density and power consumption properties of blade systems are appealing; however, the shared infrastructure between blades is generally undesirable. Blade enclosures commonly share I/O planes and network connectivity, and the blades themselves usually have little to no local storage. This is because these systems are built for compute-intensive workloads where comparatively little I/O is done. For those workloads, blade systems may be cost-effective and have a distinct advantage, but for Hadoop, they struggle to keep up.

In Worker Hardware Selection, we talked about how Hadoop prefers JBOD disk configurations. For many years—and for many systems—RAID has been dominant. There’s nothing inherently wrong with RAID; it’s fast, it’s proven, and it scales for certain types of applications. Hadoop uses disk differently. MapReduce is all about processing massive datasets, in parallel, in large sequential I/O access patterns. Imagine a machine with a single RAID-5 stripe with a stripe size of 64 KB running 10 simultaneous map tasks. Each task is going to read a 128 MB sequential block of data from disk in a series of read operations. Each read operation will be of some unknown length, dependent on the records being read and the format of the data. The problem is that even though these 10 tasks are attempting to perform sequential reads, because all I/O requests are issued to the same underlying device, the end result of interleaved reads will look like random reads, drastically reducing throughput. Contrast this with the same scenario but with 12 individual devices, each of which contains only complete 128 MB blocks. Now as I/O requests are issued by the kernel to the underlying device, it is almost certainly in the same position it was since the last read and no seek is performed. While it’s true that two map tasks could still contend for a block on a single device, the probability of that being so is significantly reduced.

Another potential pain point with RAID comes from the variation in drive rotation speed among multiple drives. Even within the same lot of drives from the same manufacturer, large variance in rotation speed can occur. In RAID, since all blocks are spread over all spindles, all operations are limited to the speed of the slowest device. In a JBOD configuration, each disk is free to spin independently and consequently, variance is less of an issue.

This brings us to shared storage systems such as SANs and NAS. Again, these systems are built with specific workloads in mind, but for Hadoop, they fall short. Keep in mind that in many ways, Hadoop specific workloads in mind, but for Hadoop, they fall short. Keep in mind that in many ways, Hadoop was created to obviate these kinds of systems. Many of these systems put a large number of fast disks behind one or two controllers with a lot of cache. Hosts are connected to the storage system either via a SAN switch or directly, depending on the configuration. The storage system is usually drastically oversubscribed; there are many more machines connected to the disks than can possibly perform I/O at once. Even with multiple controllers and multiple HBAs per host, only a small number of machines can perform concurrent I/O.

6. Hadoop – Cluster Sizing

Once the hardware for the worker nodes has been selected, the next obvious question is how many of those machines are required to complete a workload. The complexity of sizing a cluster comes from knowing—or more commonly, not knowing—the specifics of such a workload: its CPU, memory, storage, disk I/O, or frequency of execution requirements. Worse, it's common to see a single cluster support many diverse types of jobs with conflicting resource requirements. Much like a traditional relational database, a cluster can be built and optimized for a specific usage pattern or a combination of diverse workloads, in which case some efficiency may be sacrificed.

There are a few ways to decide how many machines are required for a Hadoop deployment. The first, and most common, is sizing the cluster based on the amount of storage required. Many clusters are driven by high data ingest rates; the more data coming into the system, the more machines required. It so happens that as machines are added to the cluster, we get compute resources in addition to the storage capacity. Given the earlier example of 1 TB of new data every day, a growth plan can be built that maps out how many machines are required to store the total amount of data. It usually makes sense to project growth for a few possible scenarios. For instance

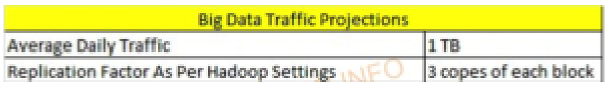
Table 1 Sample cluster growth plan based on storage

Average daily ingest rate	1 TB	
Replication factor	3 (copies of each block)	
Daily raw consumption	3 TB	Ingest × replication
Node raw storage	24 TB	12 × 2 TB SATA II HDD
MapReduce temp space reserve	25%	For intermediate MapReduce data
Node-usable raw storage	18 TB	Node raw storage – MapReduce reserve
1 year (flat growth)	61 nodes[a]	Ingest × replication × 365 / node raw storage
1 year (5% growth per month[b])	81 nodes[a]	
1 year (10% growth per month)	109 nodes[a]	

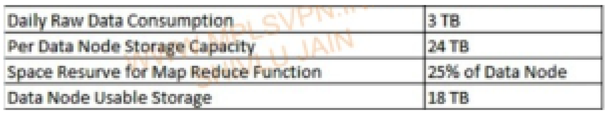
There's a clear chicken and egg problem; a job must be run with a subset of the data in order to understand how many machines are required to run the job at scale. An interesting property of MapReduce jobs is thatmap tasks are almost always uniform in execution. If a single map task takes one minute to execute and consumes some amount of user and system CPU time, some amount of RAM and some amount of I/O, 100 map tasks will simply take 100 times the resources. Reduce tasks, on the other hand, don't have this property. The number of reducers is defined by the developer rather than being based on the size of the data,

7. Detailed Case Study – Datanode Sizing

data and data node capacity projections will be given till 2017. Below are the assumptions which have been considered while capacity planning hadoop cluster:

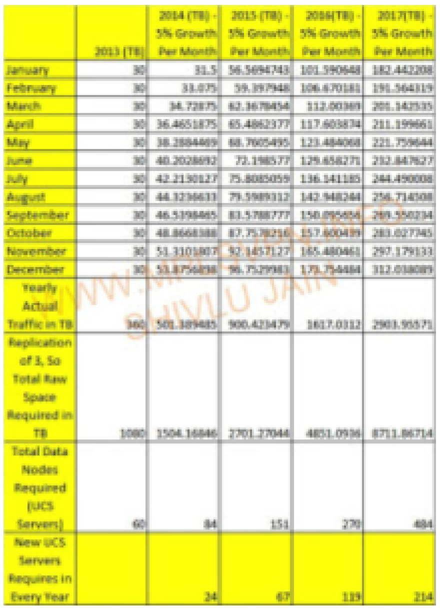


Big Data Traffic Projections	
Average Daily Traffic	1 TB
Replication Factor As Per Hadoop Settings	3 copes of each block



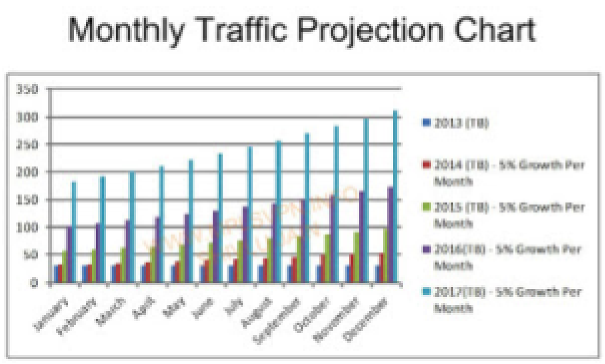
Daily Raw Data Consumption	3 TB
Per Data Node Storage Capacity	24 TB
Space Resurve for Map Reduce Function	25% of Data Node
Data Node Usable Storage	18 TB

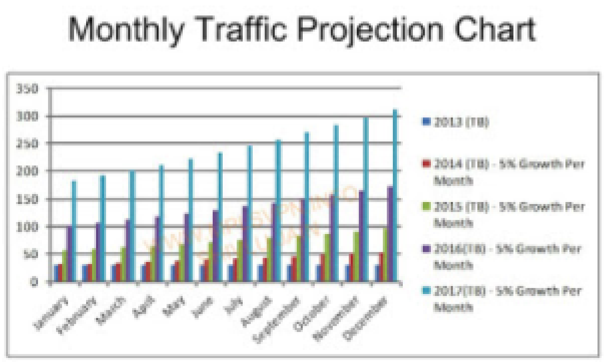
As per the above listed assumptions, starting from 1TB of dailiy data from 2013. for capacity building assuming 5% data growth per month starting from 2014 onwards. In 2013 we have 1080TB of data and by the end of 2017 we have 8711Tb of data. Almost 8 times from the starting year. Below is the table which tells how many UCS Servers(Data Nodes) will be required to handle 8711TB of data

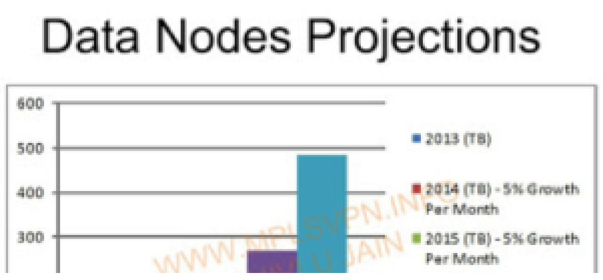


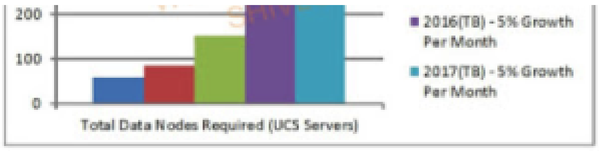
	2013 (TB)	2014 (TB) - 5% Growth Per Month	2015 (TB) - 5% Growth Per Month	2016(TB) - 5% Growth Per Month	2017(TB) - 5% Growth Per Month
January	30	31.5	56.5694743	101.590648	182.442208
February	30	33.075	59.397948	106.670181	191.564319
March	30	34.72875	62.3678454	112.00369	201.142535
April	30	36.4651875	65.4862377	117.603874	211.199661
May	30	38.2884469	68.7605495	123.484068	221.759644
June	30	40.2028692	72.198577	129.658271	232.847627
July	30	42.2130127	75.8085059	136.141185	244.490008
August	30	44.3236633	79.5989312	142.948244	256.714508
September	30	46.5398465	83.5788777	150.095656	269.550234
October	30	48.8668388	87.7578216	157.600439	283.027745
November	30	51.3101807	92.1457127	165.480461	297.179133
December	30	53.8756898	96.7529983	173.754484	312.038089
Yearly Actual Traffic in TB	360	501.389485	900.423479	1617.0312	2903.95571
Replication of 3, So Total Raw Space Required in TB	1080	1504.16846	2701.27044	4851.0936	8711.86714
Total Data Nodes Required (UCS Servers)	60	84	151	270	484
New UCS Servers Requires in Every Year		24	67	119	214

Graphical View of the projections:









8. Some important aspects of the Hadoop architecture

a. The distributed computation

At his heart, Hadoop is a distributed computation platform. This platform's programming model is Map Reduce.

In order to be efficient, Map Reduce has two prerequisites:

- Datasets must be splitable in smaller and independant blocks
- Data locality: means that the code must be moved where the data lies, not the opposite.

The first prerequisite depends on both the type of input data which feeds the cluster and what we want to do with it.

The second prerequisite involves having a distributed storage system which exposes where exactly data is stored and allows the execution of code on any storage node.

This is where HDFS is useful.

Hadoop is a Master / Slave architecture:

- The JobTracker (ResourceManagerin Hadoop 2)

ü Monitor jobs that are running on the cluster

ü Needs a lot of memory and CPU (memory bound and cpu bound)

- The TaskTracker (NodeManager + ApplicationMasterin Hadoop 2)

ü Runs tasks of a jobs on each node of the cluster. Which means Maps and Reduces

ü Its jobs need a lot of memory and CPU (memory bound and cpu bound)

The critical component in this architecture is the JobTracker/ResourceManager.

b. The distributed storage

HDFS is a distributed storage filesystem. It runs on top of another filesystem like ext3 or ext4.

In order to be efficient, HDFS must satisfy the following prerequisites:

Hard drives with a high throughput

- An underlying filesystem which supports the HDFS read and write pattern: one big read or write at a time (64MB, 128MB or 256MB)
- Network fast enough to cope with intermediate data transfer and block replication

HDFS is a Master / Slave architecture:

- The NameNode and the Secondary NameNode

ü Stores the filesystem meta informations (directory structure, names, attributes and file localization) and ensures that blocks are properly replicated in the cluster

ü It needs a lot of memory (memory bound)

- The DataNode

ü Manages the state of an HDFS node and interacts with its blocks

ü Needs a lot of I/O for processing and data transfer (I/O bound)

The critical components in this architecture are the NameNode and the Secondary NameNode.

9. Some important technical's facts to plan a cluster

No need to be an Hadoop expert but the following few facts are good to know when it comes to cluster planning. –

a. How HDFS manages its files

HDFS is optimized for the storage of large files. You write the file once and access it many times.

In HDFS, a file is split into several blocks. Each block is asynchronously replicated in the cluster. Therefore, the client sends its files once and the cluster takes care of replicating its blocks in the background.

A block is a contiguous area, a blob of data on the underlying filesystem, Its default size is 64MB but it can be extended to 128MB or even 256MB, depending on your needs.

The block replication, which has a default factor of 3, is useful for two reasons:

- Ensure data recovery after the failure of a node. Hard drives used for HDFS must be configured in JBOD, not RAID
- Increase the number of maps that can work on a bloc during a MapReduce job and therefore speedup processing

From a network standpoint, the bandwith is used at two moments:

- During the replication following a file write
- During the balancing of the replication factor when a node fails

10.How the NameNode manages the HDFS cluster

The NameNode manages the meta information of the HDFS cluster.

This includes meta informations (filenames, directories, …) and the location of the blocks of a file.

The filesystem structure is entirely mapped into memory.

In order to have persistence over restarts, two files are also used:

- a fsimage file which contains the filesystem metadata
- the edits file which contains a list of modifications performed on the content of fsimage.

The in memory image is the merge of those two files.

When the NameNode starts, it first loads fsimage and then applies the content of edits on it to recover the latest state of the filesystem.

An issue would be that over time, the edits file keeps growing undefinitely and ends up by:

- consuming all disk space
- slowdown restarts

The Secondary NameNode role is to avoir this issue by regularly merging edits with fsimage, thus pushing a new fsimage and resetting the content of edits.

The trigger for this compaction process is configurable. It can be:

- The number of transactions performed on the cluster
- The size of the edits file
- The elapsed time since the last compaction

The following formula can be applied to know how much memory a NameNode needs:

<needed memory> = <total storage size in the cluster in MB> / <Size of a block in MB> / 1000000

In other words. a rule of thumb is to consider that a NameNode needs about 1GB / 1 million blocks.

-- DataNode specific needs

11.How to determine sizing needs?

a. Determine storage needs

Storage needs are split into three parts:

- Shared needs
- NameNode and Secondary NameNode specific needs
- DataNode specific need

b. Shared needs

Shared needs are already known since it covers:

- The OS partition
- The OS logs partition

c. NameNode and Secondary NameNode specific needs

The Secondary NameNode must be identical to the NameNode. Same hardware, same configuration.

A 1TB partition should be dedicated to files written by both the NameNode and the Secondary NameNode. This is large enough so you won't have to worry about disk space as the cluster grows.

If you want to be closer to the actual occupied size, you need to take into account the parameters of the NameNode we explained above (a combination of the trigger for the compaction, the maximum fsimage size and the edits size) and to multiply this result by the number of checkpoints you want to be retained.

In any case, the NameNode must have an NFS mount point to a secured storage among its fsimage and edits directories. This mount point has the same size than the local partition for fsimage and edits mentioned above. The storage of the NameNode and the Secondary NameNode is typically performed on RAID configuration.

d. DataNode specific needs

Hardware requirements for DataNodes storage is:

ü SAS 6Gb/s controller configured in JBOD (Just a Bunch of Disk)

ü SATA II 7200 rpm hard drives between 1Tb and 3Tb

Do not use RAID on a DataNode. HDFS provides its own replication mecanism. The number of hard drive can vary depending on the total desired storage capacity.A good way to determine the latter is to start from the planned data input of the cluster.It is also important to note that for every disk, 30% of its capacity is reserved to non HDFS use.

Let's consider the following hypothesis:

ü Daily data input: 100Gb

ü HDFS replication factor: 3

ü Non HDFS reserved space per disk: 30%

ü size of a disk: 3Tb

With these hypothesis, we are able to determine the storage needed and the number of DataNodes.

Therefore we have:

ü Storage space used by daily data input : <daily data input> * <replication factor> = 300GB ü Size of a hard drive dedicated to HDFS : <Size of the hard drive > * (1 – <Non HDFS reserved space per disk>) = 2.1TB

ü Number of DataNodes after 1 year (no monthly growth) : <storage space used by daily data input> * 365 / <HDFS reserved space in a disk> = 100TB / 2.1TB = 48 DataNodes

Two important elements are not included here:

ü The monthly growth of the data input

ü The ratio of data generated by jobs processing a data input

These information depend on the needs of your business units and it must be taken into account in order to determine storage needs.

e. Determine your CPU needs

On both NameNode and Secondary NameNode, 4 physical cores running at 2Ghz will be enough. For DataNodes, two elements help you to determine your CPU needs:

- The profile of the jobs that are going to run
- The number of jobs you want to run on each DataNode

Job profile

Roughly, we consider that a DataNode can perform two kind of jobs: I/O intensive and CPU intensive.

I/O intensive jobs

These jobs are I/O bound.

For example:

ü indexing

ü search

ü clustering

ü decompression

ü data import/export

Here, a CPU running between 2Ghz and 2.4Ghz is enough.

CPU intensive jobs

These jobs are CPU bound.

For example:

ü machine learning

ü statistics

ü semantic analysis

ü language analysis

Here, a CPU running between 2.6Ghz and 3Ghz is enough.

The number of jobs

The number of physical cores determine the maximum number of jobs that can run in parallel on a DataNode. It is also important to keep in mind that there is a distribution between Map and Reduce tasks on DataNodes (typically 2/3 Maps and 1/3 Reduces).

To determine you needs, you can use the following formula:

(<number of physical cores> – 1) * 1.5 = <maximum number of tasks>

or, if you prefer to start from the number of tasks and adjust the number of cores according to it:

(<maximum number of tasks> / 1.5) + 1 = <number of physical cores>

The number 2 keeps 2 cores away for both the TaskTracker (MapReduce) and DataNode (HDFS) processes.

the number 1.5 indicates that a physical core, due to hyperthreading, might process more than one job at the same time.

f. Determine your memory needs

This is a two step process:

ü Determine the memory of both NameNode and Secondary NameNode

ü Determine the memory of DataNodes

In both cases, you should use DDR3 ECC memory.

Determine the memory of both NameNode and Secondary NameNode –

As explained above the NameNode manages the HDFS cluster metadata in memory. The memory needed for the NameNode process and the memory needed for the OS must be added to it.

The Secondary NameNode must be identical to the NameNode. Given these informations

we have the following formula:

<Secondary NameNode memory> = <NameNode memory> = <HDFS cluster management memory> + <2GB for the NameNode process> + <4GB for the OS

Determine the memory of DataNodes

The memory needed for a DataNode is determined depending on the profile of jobs which will run on it.

For I/O bound jobs, between 2GB and 4GB per physical core.

For CPU bound jobs, between 6GB and 8GB per physical core.

In both cases, the following must be added:

2GB for the DataNode process which is in charge of managing HDFS blocks

2GB for the TaskTracker process which is in charge of managing running tasks on the node 4GB for the OS

Which leads to the following formulas:

<DataNode memory for I/O bound profile> = 4GB * <number of physical cores> + <2GB for the DataNode process> + <2GB for the TaskTracker process> + <4GB for the OS>

<DataNode memory for CPU bound profile> = 8GB * <number of physical cores> + <2GB for the DataNode process> + <2GB for the TaskTracker process> + <4GB for the OS>

g. Determine your network needs

The two reasons for which Hadoop generates the most network traffic are:

- The shuffle phase during which Map tasks outputs are sent to the Reducer tasks
- Maintaining the replication factor (when a file is added to the cluster or when a DataNode is lost)

In spite of it, network transfers in Hadoop follow an East/West pattern which means that even though orders come from the NameNode, most of the transfers are performed directly between DataNodes.

As long as these transfers do not cross the rack boundary, it is not a big issue and Hadoop does its best to perform only such transfers.

However, inter-rack transfers are sometimes needed, for example for the second replica of an HDFS block.

This is complex subject but as a rule of thumb, you should:

- Use a Spine Fabric network topology

ü Better throughput

ü Better resiliency to failures

- Avoid oversubscribtion of switches

ü A 1Gb 48 ports top of rack switch must have 5 ports at 10Gb on distribution switches

ü Avoids network slowdown for a cluster under heavy load (jobs + data input)

- If the cluster is I/O boundor if you plan to perform data input on recurrent data input which sature the 1GB and which cannot be performed outside of office hours, you should use:

ü 10Gb Ethernet intra rack

ü N x 10Gb Ethernet inter rack

- If the cluster is CPU bound, you should use:

ü 2 x 1Gb Ethernet intra rack

ü 10Gb Ethernet inter rack

12.Some other important questions and considerations as you get started with Hadoop.

- How will multi-tenancy and sharing work if more than one group is going to be using your cluster.
- Should I have one or a few big Hadoop clusters, or many small clusters
- Understand your storage, processing, and concurrency needs. Not all Hadoop schedulers are created equal for all situations.
- Do you need or want to leverage virtualization and or cloud bursting?
- Choose your hardware carefully to keep costs per TB low. How to mange TB vs cpu/core is important.
- Understand what you need in your edge nodes for utility and add-on software.
- Plan your data acquisition and export needs between your Hadoop cluster and the rest of your ecosystem.
- Understand your security needs at a data and functional level.
- Why they are pursuing Big Data (other than it is the hot thing to do).
- How Hadoop differs from past propriety Big Data solutions.
- How it can fit along side existing legacy systems.
- How to ultimately manage costs and expectations at both a management and technical level.
- What are your up time requirements? Plan for rolling patches and upgrades

Article is in Development Process. If there are any remarks please contact: olena.khokhlova@software.com.pl

Pre Order: goo.gl/i07T1K

[/private]