International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-2, Issue-9 E-ISSN: 2347-2693

The Hadoop Dispersed File system: Balancing Movability And Performance

K.Udhaya Malar[1*], D.Ragupathi[2] and G.M.Prabhu[3th]

[18,2]*Department of Computer Science, AVVM Sri Pushpam College, Bharathidasan University, India,*
[3] *R&D of Computer Science, STEPINFOTECH, India*

www.ijcaonline.org

Received: Aug/29/2014 Revised: Sep/10/2014 Accepted: Sep/25/2014 Published: Sep/30/2014

Abstract— Hadoop is a general open-source application of map reduce for the examination of big datasets. To achieve storing capitals crosswise the cluster, Hadoop uses a dispersed user-level file system. This file system — HDFS — is written in java and envisioned for movability crosswise varied hardware and software platforms. This newspaper inspects the presentation of HDFS and uncovers numerous presentation issues. First, architectural blockages be in the Hadoop application that consequence in incompetent HDFS usage owing to postponements in preparation new map reduce tasks. Second, movability limits stop the java application after abusing topographies of the innate platform. Third, HDFS implicitly brands movability molds about how the innate phase achieves storing resources, smooth however innate file systems and I/O schedulers vary widely in arrangement and behavior. This newspaper examines the root details of these presentation blockages in instruction to assess tradeoffs among movability and presentation in the Hadoop dispersed file system.

Keywords—Component; Formatting; Style; Styling; Insert (Key Words)

I. INTRODUCTION

The assimilation of calculating hooked on our daily lives is allowing the cohort of figures at unprecedented rates. In 2008, IDC projected that the "digital universe" incomplete 486 Exabyte of figures [2]. The map reduce software design faultless has emerged as a climbable way to do data-intensive computations on creation bunch processers [8], [9]. The achievement of map reduce has enthused the formation of Hadoop, a general open-source implementation. Written in java for cross-platform portability, Hadoop is working today by a wide variety of profitable and theoretic users for backend figures processing. A key constituent of Hadoop is the Hadoop dispersed folder scheme (HDFS), which is used to store all input and production figures for applications.

The competence of the map reduce faultless has remained questioned in new investigation contrasting it with the alike catalogue example for large-scale figures analysis. Typically, Hadoop is used as representative of the map reduce faultless since branded (e.g., Google-developed) applications with possibly advanced presentation are not openly available. In one study, Hadoop presentations did ill in trials when likened to alike agendas using alike catalogues [18], [22]. However, this work did not do the summarizing essential to differentiate the important presentation of the map reduce software design faultless after a exact implementation. We discovery that it is essentially the application of the Hadoop storing scheme that damages presentation significantly.

This newspaper is the chief to inspect the connections among Hadoop and storage. We tag how the user-level Hadoop file system, in its home of professionally capturing the filled presentation possible of the underlying bunch hardware, essentially damages appeal presentation significantly. The exact blockages in HDFS can be categorized hooked on three categories:

Software architectural blockages — HDFS is not utilized to its filled possible owing to preparation postponements in the Hadoop building that consequence in bunch nodes waiting for new tasks. In its home of using the diskette in a flowing manner, the admittance design is periodic. Further, smooth when tasks are obtainable for computation, the HDFS customer code, chiefly for folder reads, serializes calculation and I/O in its home of decoupling and pipelining those operations. Figures prefetching is not working to recuperate performance, smooth however the typical map reduce flowing admittance design is extremely predictable.

Portability limits — certain performance-enhancing topographies in the innate file system are not obtainable in java in a platform-independent manner. This comprises choices such as avoiding the file system sheet accumulation and moving figures straight after diskette hooked on user buffers. As such, the HDFS application runs less professionally and has advanced computer usage than would then be necessary.

Portability molds — the classic idea of software movability is simple: fixes the appeal run on manifold platforms? But, a broader idea of movability is: fixes the appeal do well on manifold platforms? While HDFS is strictly portable, its presentation is extremely reliant on on the conduct of underlying software layers, exactly the os i/o scheduler and innate filesystem allocation algorithm.

Here, we count the influence and significance of these HDFS bottlenecks. Further, we explore possible answers and inspect how they influence movability and performance. these answers comprise better i/o scheduling, adding pipelining and prefetching to composed task preparation and HDFS clients, preallocating folder interplanetary on disk, and

Corresponding Author: *K.Udhaya Malar*

adapting or eliminating the local filesystem, amid additional methods.

MapReduce schemes such as Hadoop are used in largescale deployments. eliminating HDFS blockages will not only development appeal performance, nonetheless also recuperate general bunch efficiency, thereby plummeting control and cooling prices and permitting additional calculation to be accomplished with the alike amount of bunch nodes.

In this paper, unit ii tags Hadoop and its dispersed filesystem, while unit iii characterizes its current performance. Unit IV deliberates possible presentation developments to Hadoop and their movability implications. Unit V deliberates linked work, and unit VI accomplishes this paper.

II. BACKGROUND

Hadoop [3] is an exposed basis outline that gears the mapreduce alike software design faultless [8]. Hadoop is calm of a mapreduce engine and a user-level filesystem that achieves storing capitals crosswise the cluster. For movability crosswise a variety of stages — Linux, FreeBSD, mac OS/X, Solaris, and Windows — composed devices are written in java and only need creation hardware.

MapReduce Engine

In the mapreduce model, calculation is alienated hooked on a map drive and a decrease function. The map drive takes a key/value couple and crops one or additional central key/value pairs. The decrease drive then takes these central key/value couples and combines all values consistent to a single key. The map drive can run self-sufficiently on all key/value pair, skimpy enormous quantities of parallelism. Similarly, the decrease drive can run self-sufficiently on all central key, also skimpy important parallelism.

In Hadoop, a central Job-tracker facility is accountable for splitting the input figures hooked on parts for dispensation by self-governing map and decrease tasks, preparation all task on a bunch node for execution, and recovering after disappointments by re-running tasks. On all node, a task tracker facility runs mapreduce tasks and occasionally contacts the Job-tracker to bang task completions and appeal new tasks. By default, when a new task is received, a new jvm example will be spawned to do it.

Hadoop dispersed folder System

The Hadoop dispersed folder scheme (HDFS) delivers worldwide admittance to annals in the bunch [4], [23]. For all-out portability, HDFS is applied as a user-level filesystem in java which feats the innate filesystem on all node, such as ext3 or NTFS, to store data. Annals in HDFS are alienated hooked on big blocks, typically 64MB, and all part is stowed as a distinct folder in the local filesystem.

HDFS is applied by two services: the Name-node and DataNode. the Name-node is accountable for upholding the HDFS almanac tree, and is a central facility in the bunch working on a single node. clienteles communication the Name-node in instruction to do communal filesystem operations, such as open, close, rename, and delete. the Name-node fixes not store HDFS figures itself, nonetheless somewhat upholds a mapping among HDFS folder name, a list of parts in the file, and the DataNode(s) on which those parts are stored.

In addition to a central NameNode, all residual bunch nodes deliver the Data-node service. All Data-node supplies HDFS parts on behalf of local or distant clients. All part is protected as a distinct folder in the node's local filesystem. Since the Data-node abstracts absent particulars of the local storing arrangement, all nodes do not consume to use the alike local filesystem. Parts are shaped or destroyed on Data-nodes at the appeal of the NameNode, which validates and procedures needs after clients. While the Name-node achieves the namespace, clienteles attach straight with Data-nodes in instruction to read or write figures at the HDFS part level.

Hadoop mapreduce presentations use storing in a way that is dissimilar after general-purpose calculating [11]. First, the figures annals protected are large, typically tens to hundreds of gigabytes in size. Second, these annals are operated via flowing admittance designs typical of batch-processing workloads. When interpretation files, big figures sections (several hundred kilobytes or more) are protected per operation, with consecutive needs after the alike customer iterating through a folder area sequentially. Similarly, annals are also written in a consecutive manner.

This emphasis on flowing workloads is clear in the arrangement of HDFS. First, a humble constancy faultless (write-once, read-many) is used that fixes not allow figures to be adapted once written. this is well right to the flowing admittance design of board applications, and recovers bunch climbing by simplifying organization requirements. Second, all folder in HDFS is alienated hooked on big parts for storing and access, typically 64MB in size. Portions of the folder can be stowed on dissimilar bunch nodes, complementary storing capitals and demand. Working figures at this granularity is well-organized since streaming-style presentations are probable to read or write the whole part beforehand touching on to the next. In addition, this arrangement excellent recovers presentation by lessening the quantity of metadata that necessity be followed in the filesystem, and permits admittance dormancy to be amortized over a big volume of data. Thus, the filesystem is enhanced for tall Band-width in its home of low latency. This permits non-interactive presentations to procedure figures at the fastest rate.

To read an HDFS file, customer presentations just use a normal java folder input stream, as if the folder was in the innate filesystem. Behindhand the scenes, however, this watercourse is operated to but figures after HDFS instead. First, the Name-node is contacted to appeal admittance permission. If granted, the Name-node will translate the HDFS filename hooked on a list of the HDFS part ids comprising that folder and a list of Data-nodes that store all block, and reappearance the lists to the client. Next, the customer unbolts a joining to the "closest" Data-node (based on Hadoop rack-awareness, nonetheless optimally the alike node) and needs an exact part ID. That HDFS part is repaid

over the alike connection, and the figures brought to the application.

To write figures to HDFS, customer presentations see the HDFS folder as a normal production stream. Internally, however, watercourse figures is chief disjointed hooked on HDFS-sized parts (64MB) and then lesser packs (64kB) by the customer thread. All pack is enquired hooked on a FIFO that can grip up to 5MB of data, thus decoupling the appeal thread after storing scheme dormancy through usual operation. A additional thread is accountable for DE queuing packs after the FIFO, coordinating with the Name-node to assign HDFS part ids and destinations, and

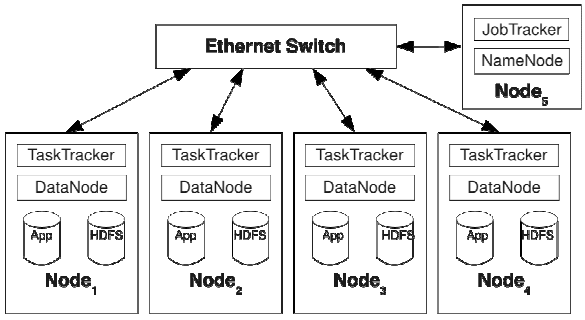


Fig. 1.Cluster Setup

Transmitting parts to the Data-nodes (either local or remote) for storage. A third thread achieves acknowledgements after the Data-nodes that figures has remained dedicated to disk.

HDFS Replication

For reliability, HDFS gears an involuntary repetition system. By default, two reproductions of all part are stowed by dissimilar Data-nodes in the alike rack and a third reproduction is stowed on a Data-node in a dissimilar rack (for better reliability). Thus, in usual bunch operation, all Data-node is servicing composed local and distant clienteles simultaneously. HDFS repetition is transparent to the customer application. When writing a block, a pipeline is recognized whereby the customer only communicates with the chief DataNode, which then echoes the figures to a additional DataNode, and so on, until the wanted amount of replicas consume remained created. The part is only finished when all nodes in this repetition pipeline consume positively dedicated all figures to disk. Data-nodes occasionally bang a list of all parts stowed to the NameNode, which will verify that all folder is adequately replicated and, in the circumstance of failure, instruct Data-nodes to brand supplementary copies.

III. PRESENTATION CHARACTERIZATION

In this section, the Hadoop dispersed filesystem is assessed in instruction to classify blockages that damage appeal performance.

A. New Setup

For presentation characterization, a 5-node Hadoop bunch was configured, as exposed in figure 1. The chief 4 nodes if composed calculation (as mapreduce clients) and storing capitals (as Data-node servers), and the 5th node helped as composed the mapreduce scheduler and Name-node storing manager. All node was a 2-processor Opteron waiter running at 2.4 ghz or above with 4GB of ram and a gigabit Ethernet NIC. All nodes used FreeBSD 7.2, Hadoop outline 0.20.0, and java 1.6.0. The chief four nodes were prearranged with two Seagate Barracuda 7200.11 500GB firm drives. One diskette stowed the working system, Hadoop application, and appeal scratch space, while the additional diskette stowed only HDFS data. All disks used the evasion UFS2 filesystem for FreeBSD with a 16kB part size and 2kB fragment size. Unless then stated, Hadoop repetition was disabled.

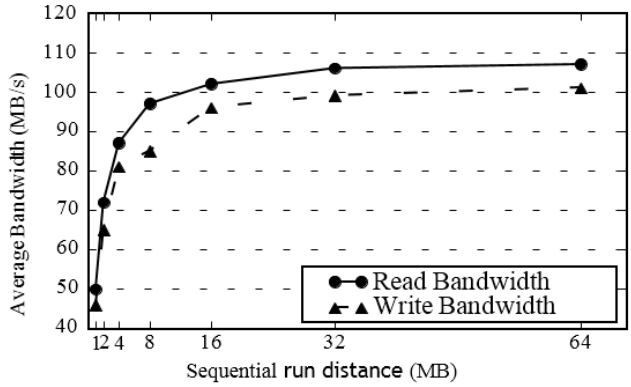


Fig. 2. raw firm drive read and Write Band-width after aio test With chance pursue each n Megabytes

To tag the Hadoop framework, a variety of test presentations were installed as exposed in bench I. this test set comprises a humble HDFS artificial writer and booklover responsibility consecutive flowing access, an HDFS writer that brands chance two figures or text strings and writes them to the diskette in a consecutive fashion, a humble amount sort, and a humble hunt for an infrequent text design in a big file. Hadoop is still a young platform, and the insufficient multifaceted presentations used in manufacturing are branded and thus unavailable. For judgment purposes, a package written in c was used to do asynchronous i/o (AIO) on the raw diskette to control the best-case performance, self-governing of any Hadoop, Java, or filesystem-specific overheads.

B. Raw Diskette Performance

To home an upper sure on Hadoop performance, the raw Band-width of the creation firm drive used in the bunch was measured. To count the presentation influence of seeks, the aio package (running on a raw disk, not confidential Hadoop) was prearranged to do long duration consecutive reads and writes, with a pursue to a chance associated site each n megabytes. This signifies the best-case Hadoop conduct where a big HDFS part of n megabytes is streamed after disk, and then the drive pursues to a dissimilar site to but additional big block. The outer districts of the drive (identified by low rational addresses) were used to get top bandwidth. As

exposed in figure 2, the drive presentation approaches its top Band-width when pursues happen less frequently than once each 32MB of consecutive figures accessed. Thus, the HDFS arrangement choice to use big 64MB parts is fairly sensible and, presumptuous that the filesystem upholds folder contiguity, must allow tall diskette bandwidth.

C. *Software Architectural Bottlenecks*

Hadoop appeal presentation agonizes owing to architectural blockages in the way that presentations use the Hadoop filesystem. Ideally, mapreduce presentations must manipulate the diskette using flowing admittance patterns. The appeal outline must allow for figures to be read or written to the diskette continuously, and overlay calculation with I/O. frequent humble presentations with low calculation supplies do not achieve this faultless working mode. Instead, they utilize the diskette in an episodic fashion, lessening performance.

Code	Program	Data Size	Notes
S-Wr	Synthetic Write	10GB / node	Hadoop consecutive write
S-Rd	Synthetic Read	10GB / node	Hadoop consecutive read
Rnd-Text	Random text Writer	10GB / node	Hadoop consecutive write
Rnd-Bin	Random two Writer	10GB / node	Hadoop consecutive write
Sort	Simple Sort	40GB / cluster	Hadoop sort of amount data
Search	Simple Search	40GB / cluster	Hadoop hunt of text figures for infrequent string
AIO-Wr	Synthetic Write	10GB / node	Native c package - asynchronous I/O
AIO-Rd	Synthetic Read	10GB / node	Native c package - asynchronous I/O

TABLE i appeal test SUITE

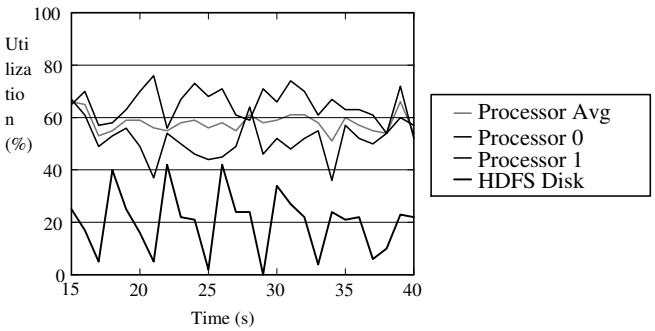


Fig. 3.Simple hunt computer and diskette Utilization (% of time diskette Had 1 or additional unresolved Requests)

The conduct of the diskette and computer utilization over time for the humble hunt normal is exposed in figure 3. Diskette utilization was sluggish as the fraction of time that the diskette consumed at minimum one I/O appeal outstanding. This summarizing did not amount the comparative competence of diskette admissions (which is prejudiced by dangerous pursues and appeal size), nonetheless just inspected whether or not the diskette was reserved adequately eventful with unresolved facility requests. Here, the scheme is not opening the diskette in a incessant flowing chic as desired, smooth however there are plenty computer capitals still available. Rather, the scheme is interpretation figures in bursts, dispensation it (by penetrating for a short text cord in all input line), and then fetching additional figures in an episodic manner. This conduct is also clear in additional presentations such as the sort benchmark, not exposed here.

The general scheme influence of this episodic conduct is exposed in figure 4, which gifts the even HDFS diskette and computer utilization for all appeal in the test suite. the aio test agendas (running as innate applications, not in Hadoop) reserved the diskette saturated with i/o needs closely all the time (97.5%) with very low computer utilization (under 3.5%). certain Hadoop agendas (such as s-wr and Rnd-Bin) also reserved the diskette equivalently busy, albeit at abundant advanced computer usage owing to Hadoop and java virtual machine overheads. in contrast, the residual agendas consume deprived reserve utilization. for instance, the hunt package admissions the diskette less than 40% of the time, and uses the computers less than 60% of the time.

This deprived competence is a consequence of the way presentations are arranged in Hadoop, and is not a blockage shaped by HDFS. By default, the test presentations alike hunt and sort were alienated hooked on hundreds of map tasks that all procedure only a single HDFS part or less beforehand exiting. This can haste recovery after node disappointment (by plummeting the quantity of work lost) and abridge bunch scheduling. It is informal to take a map task that admissions a single HDFS part and assign it to the node that covers the data. Preparation develops additional difficult, however, when map tasks admittance an area of manifold HDFS blocks, all of which forte exist in on dissimilar nodes. Unfortunately, the welfares of using a big amount of minor tasks originate with a presentation value that is chiefly tall for presentations alike the hunt test that whole tasks quickly. When a map task completes, the node can be indolent for numerous instants until the task tracker polls the Job-tracker for additional tasks. By default, the least polling intermission is 3 instants for a minor cluster, and upsurges with bunch size. Then, the Job-tracker runs a preparation procedure and revenues the next task to the Task Tracker. Finally, a new java virtual machine (JVM) is started, after which the node can recommence appeal processing.

This blockage is not shaped by the filesystem, nonetheless fixes touch how the filesystem is used. cumulative the HDFS part size to 128MB, 256MB, or advanced — a commonly-proposed optimization [17], [18] — indirectly recovers presentation not since it alleviates any inefficiency in HDFS nonetheless since it decreases the incidence at which a node is indolent and awaiting scheduling. Additional option, over-subscribing the bunch by transmission frequent additional map and decrease tasks than there are computers and disks in the bunch nodes, may also alleviate this problematic by meeting calculation and I/O after dissimilar tasks. But, this method risks humiliating presentation in a dissimilar way by cumulative I/O disagreement after manifold clients, a problematic deliberated additional in unit III-E.

To additional straight incidence the presentation bottleneck, Hadoop can be prearranged to re-use the alike JVM for manifold tasks in its home of preliminary a new JVM all time. In the hunt test, this augmented presentation by 27%, while diskette utilization was still underneath 50%. Further, the quantity of work complete by all map task can be adjusted. creation all map task procedure 5GB of figures in its home of 64MB beforehand departing better hunt presentation by 37% and boosted diskette utilization to over 68%.

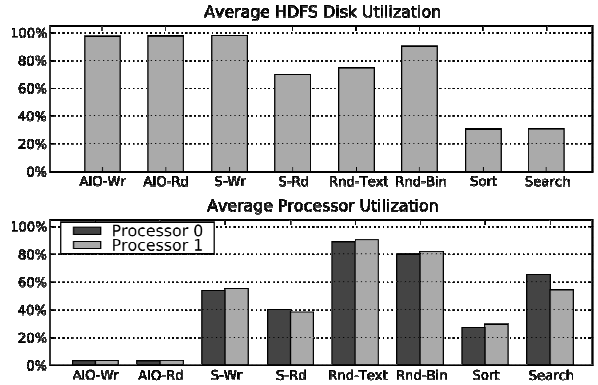


Fig. 4. Average computer and HDFS diskette Utilization (% of time diskette consumed 1 or additional unresolved Requests)

These fast examinations show that HDFS — the emphasis of this newspaper — is not the aim of this presentation bottleneck. Rather, additional work in the rest of the Hadoop outline is needed. Answers such as pipelining and prefetching tasks after the Job-tracker in advance may assistance hide preparation latency.

Even when tasks are obtainable for dispensation and all task is working over manifold HDFS parts situated on the alike node, a blockage still is since the HDFS customer application is extremely serialized for figures reads. as deliberated in unit II, there is no pipelining to overlay appeal calculation with I/O. the appeal necessity wait on the i/o scheme to communication the NameNode, communication the DataNode, and transfer figures beforehand processing. This dormancy is better on big bunches with eventful NameNodes, or in circumstances where the figures being protected is not on the alike node. Similarly, the I/O scheme necessity wait for the appeal to whole dispensation beforehand getting additional request. Outside the absence of pipelining, there is also no figures prefetching in the system, notwithstanding the detail that mapreduce presentations admittance figures in a foreseeable flowing fashion. Only metadata is prefetched, exactly the mapping among HDFS filename and part IDs. Somewhat than communication the Name-node all time a new part id is required, the customer caches the next 10 parts in the folder with all request.

D. Movability Limitations

The Hadoop outline and filesystem impose a important computer above on the cluster. While certain of this above is characteristic in if essential functionality, additional above is experienced owing to the arrangement goalmouth of creation a moveable mapreduce implementation. These are mentioned to as movability Limitations.

An example of the total above experienced is exposed in figure 4. The asynchronous I/O write (AIO-Wr) test package — written in c and opening the raw diskette self-governing of the filesystem — takes less than 10% of the computer through operation. But, the artificial writer (S-Wr) test package — written in java and running in Hadoop — takes over 50% of the computer to write figures to diskette in a alike chic with equal bandwidth. That above originates after four places: Java, HDFS implementation, the local filesystem, and the filesystem sheet cache. While the chief two expenses are characteristic in the Hadoop implementation, the latter two are not.

As deliberated in unit II, the Hadoop Data-node uses a local filesystem to store data, and all HDFS part is as a distinct folder in the innate filesystem. While this method brands Hadoop humble to install and portable, it executes a calculation above that is current irrespective of the exact filesystem used. The filesystem takes computer time to brand figures allocation and assignment decisions, while the filesystem sheet accumulation consumes composed computer and memory capitals to manage.

To count the computer capitals consumed by the filesystem and cache, an artificial java package was used to read and write 10GB annals to diskette in a flowing chic using 128kB buffered blocks. The test package incurs folder admittance expenses imposed by java nonetheless not any Hadoop-specific overheads. It was did composed on a raw diskette and on a big folder in the filesystem in instruction to liken the above of composed approaches. kernel callgraph summarizing was used to excellence above to exact os functions.

As exposed in bench II, using a filesystem has a low computer overhead. When reading, 4.4% of the computer time was consumed handling filesystem and folder accumulation linked functions, and while writing, 7.2% of the computer time was consumed on the alike kernel tasks. This above would be inferior if supplementary or earlier computers consumed remained used for the new cluster, and advanced if supplementary or earlier disks were added to the cluster.

Metric	Read		Write	
	Raw	Filesystem	Raw	Filesystem
Bandwidth (MB/s)	99.9	98.4	98.1	94.9
Processor (total)	7.4%	13.8%	6.0%	15.6%
Processor (FS+cache)	N/A	4.4%	N/A	7.2%

TABLE II

PROCESSOR above of diskette as raw expedient VERSUS diskette WITH filesystem and sheet accumulation (FS+CACHE)

E. Movability Assumptions

A latter lesson of presentation blockages is in the Hadoop filesystem that we mention to as movability Assumptions. Specifically, these blockages be since the HDFS application brands implicit molds that the underlying os and filesystem

will behave in a best way for Hadoop. Unfortunately, I/O schedulers can aim dangerous pursues under simultaneous workloads, and diskette allocation procedures can aim dangerous fragmentation, composed of which damage HDFS presentation significantly. These agents are outside the straight switch of HDFS, which runs confidential a java virtual machine and achieves storing as a user-level application.

1) Scheduling: HDFS presentation damages whenever the diskette is communal among simultaneous writers or readers. Dangerous diskette pursues happen that are counter-productive to the goalmouth of exploiting general diskette bandwidth. This is an important problematic that touches HDFS running on all platforms. Current I/O schedulers are envisioned for general-purpose workloads and exertion to portion capitals fairly among competing processes. In such workloads, storing dormancy is of equal rank to storing bandwidth; thus, fine-grained justice is if at a minor granularity (an insufficient hundred kilobytes or less). in contrast, mapreduce presentations are closely completely dormancy insensitive, and thus must be arranged to exploit diskette Band-width by treatment needs at a big granularity (dozens of megabytes or more).

To show deprived preparation by the working system, an artificial test package in Hadoop was used to write 10GB of HDFS figures to diskette in a consecutive flowing way using 64MB blocks. 1-4 reproductions of this appeal were run alongside on all bunch node. All example writes figures to a distinct HDFS file, thus compelling the scheme to portion incomplete I/O resources. The collective Band-width attained by all writers on a node was recorded, as exposed in figure 5(a). Collective Band-width released by 38% when touching after 1 writer to 2 simultaneous writers, and released by a supplementary 9% when a third writer was added.

This presentation squalor happens since the amount of pursues upsurges as the amount of writers upsurges and the diskette is forced to change among distinct figures streams. Eventually, non-sequential needs explanation for up to 50% of diskette accesses, notwithstanding the detail that, at the appeal level, figures is being protected in a flowing chic that must ease big HDFS-sized part admissions (e.g., 64MB). Since of these seeks, the even consecutive run distance reductions dramatically as the amount of writers increases. What was originally a 4MB even run distance reductions to less than 200kB with the addition of an additional simultaneous writer, and ultimately damages additional to about 80kB. Such short consecutive runs straight influence general diskette I/O bandwidth, as understood in figure 2.

An alike presentation topic happens when HDFS is distribution the diskette among simultaneous readers. To show this, the alike artificial test package was used. First, a single writer was used per node to write 4 distinct 10GB HDFS files. A single writer procedure creates figures that is extremely adjoining on disk, as exposed by the insignificant fraction of pursues in the preceding 1-writer test. Then, 1-4 simultaneous artificial booklover presentations were used per node to all read back a dissimilar folder after disk.

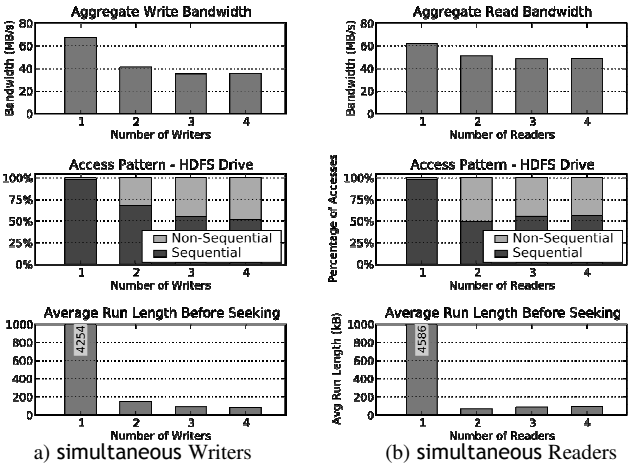
In this test, the collective Band-width for all booklovers on an exact node was recorded, as exposed in figure 5(b). The collective Band-width released by 18% when touching after 1 booklover to 2 readers. This is since the amount of pursues augmented as the amount of booklovers increased, reaching up to 50% of total diskette accesses. This also impacted the even run distance beforehand seeking, which released after over 4MB to well under 200kB as the amount of simultaneous booklovers increased.

By default, the freebsd schemes used for challenging working a humble elevator I/O scheduler. If the scheme consumed used an additional urbane scheduler that decreases seeks, such as the preventive Scheduler, this problematic may consume remained masked, and the limits of the moveable HDFS application hidden. The preventive scheduler efforts to decrease pursues by waiting a short retro after all appeal to see if additional consecutive needs are forthcoming [13]. If they are, the needs can be serviced without additional seeks; if not, the diskette pursues to facility a dissimilar client.

A humble preventive scheduler for freebsd was prearranged and verified using simultaneous examples of the Hadoop artificial writer and booklover application. The new scheduler consumed no influence on the i/o Band-width of the test programs. Summarizing exposed that, for the read workload, the scheduler did recuperate the admittance physiognomies of the drive. A tall grade of consecutive admissions (over 95%) and a big consecutive run distance (over 1.5MB) were upheld when touching after 1 to 4 simultaneous readers. But, since the drive was frequently indolent waiting on new read needs after the synchronous HDFS implementation, general appeal Band-width did not improve. Summarizing also presented that the scheduler consumed no influence on the admittance physiognomies of write workloads. This is probable since the filesystem part allocator is creation choices beforehand the I/O scheduler. Thus, smooth if the preventive scheduler waits for the next customer request, it is frequently not adjoining in this filesystem and thus not favored over any additional undecided requests.

2) Fragmentation: in addition to deprived I/O scheduling, HDFS also agonizes after folder disintegration when distribution a diskette among manifold writers. the all-out likely folder contiguity — the size of an HDFS part — is not preserved by the general-purpose filesystem when creation diskette allocation decisions.

To amount folder disintegration on a newly arranged disk, 1-4 artificial writer presentations were used per node to all brand 10GB files, written concurrently. Next, a single artificial booklover appeal was used to read back one of the 1-4 annals originally created. If the figures on diskette is contiguous, the single booklover must be bright to admittance it with a least of seeks; otherwise, the folder necessity be disjointed on disk.



a) simultaneous Writers (b) simultaneous Readers

The consequences after this trial are exposed in figure 6. Here, folder disintegration happens whenever manifold writers use the diskette concurrently. When the single booklover admissions figures written when only one writer was active, it obtains tall Band-width thanks to an insignificant fraction of chance seeks, presentation that the figures was written to the diskette in big adjoining blocks. However, when the booklover admissions figures written when 2 writers were active, read Band-width droplets by 30%. The aim of this droplet is an upsurge in the amount of chance seeks, and a consistent discount in the even consecutive run distance after over 4MB to about 250kB. This trend lasts when 3-4 simultaneous writers were used, presentation that annals hurt after cumulative disintegration as the amount of simultaneous writers is increased. The equal of disintegration currently was shaped by using a newly arranged diskette for all experiment. In a Hadoop bunch running for frequent months or years, the real-world diskette disintegration would probably be greater.

The even run lengths exposed in figure 6 for the disintegration test are closely twice as long as the manifold writers test exposed in figure 5(a). This demonstrates that after a diskette fixes a pursue to facility a dissimilar writer, it will occasionally hurdle back to the preceding site to surface writing out an adjoining cluster. Unfortunately, the filesystem used only efforts to uphold minor bunches (128kB). As such, the general equal of on-disk folder contiguity is still very low likened to what would be best for HDFS.

F. Discussion

As exposed previously, simultaneous booklovers and writers damage the presentation of the Hadoop filesystem. This consequence is not a infrequent incidence in bunch procedure that can be disregarded. Simultaneous diskette admittance is originate in usual procedure since of two key elements: manifold map/reduce procedures and figures replication.

MapReduce is envisioned to allow calculation tasks to be effortlessly dispersed crosswise a big processer cluster. This alike parallelization method also permits the exploitation of manifold computer cores. In the bunch used for experimentation, all node consumed 2 processors, and thus was prearranged to run 2 map reduce procedures concurrently. While 2 procedures permissible the test set to use additional calculation resources, the simultaneous reads and writes shaped slowed the overall application time. while it forte be sensible in this formation to whichever install a additional HDFS diskette or run only 1 appeal procedure per node, this "solution" is not climbable when bunch nodes are complete with computers refuge 4, 8, 16, or additional cores. it is unreasonable to whichever install one diskette per vital or permission those centers indolent — deserting the parallelization welfares complete likely by the map reduce software design chic — to avoid presentation glitches shaped by simultaneous diskette access. Further, Hadoop installations frequently deliberately oversubscribe the bunch by running additional map or decrease tasks than there are computers or disks. This is complete in instruction to decrease scheme indolent time shaped by tall dormancy in preparation and initiating new tasks as recognized in unit III-C.

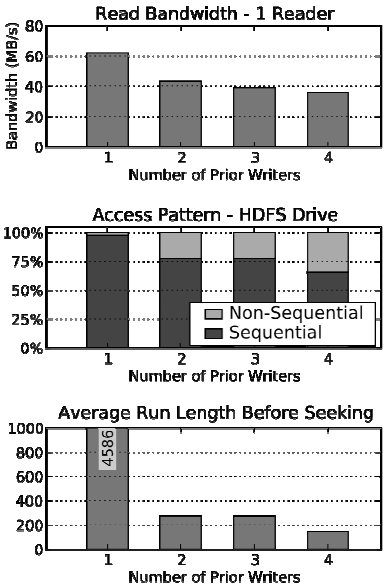


Fig. 6. One Hadoop artificial booklover package opening figures after one artificial Writer. (Data was before complete with 1-4 Concurrent Writers)

In addition to manifold calculation processes, simultaneous diskette admittance can also arise owing to HDFS figures replication. as before mentioned, bunches typically function with a repetition topic of 3 for redundancy, intelligence that one reproduction of the figures is protected locally, one reproduction is protected on additional node in the alike rack, and a third reproduction is protected on a node in a distant rack. But, writing figures to diskette after composed local and distant agendas details simultaneous diskette accesses.

The consequence of a bunch repetition topic of 2 on diskette admittance designs was tested. The consequences in bench iii show that replication is a trivial way to crop simultaneous access. The conduct of the artificial writer with repetition allowed is extremely alike to the conduct of 2 simultaneous writers, before exposed in figure 5(a). The mix of consecutive and chance diskette admissions is similar, as is the very minor even run distance beforehand seeking.

Metric	Synthetic Write	Synthetic Read
Sequential %	77.9%	70.3%
Non-Sequential %	22.1%	29.7%
Avg. Seq. run Length	275.2kB	176.8kB

TABLE III
DISK admittance physiognomies for artificial WRITE and READ APPLICATIONS WITH repetition ENABLED

Alike comments for the read test can be complete against the conduct of 2 simultaneous readers, before exposed in figure 5(b). Thus, the presentation squalor after simultaneous HDFS admittance is current in each Hadoop bunch using replication.

G. additional stages – linux and Windows

The main consequences exposed in this newspaper used HDFS on FreeBSD 7.2 with the UFS2 filesystem. For judgment purposes, HDFS was also verified on Linux 2.6.31 using the ext4 and XFS filesystems and Windows 7 using the NTFS filesystem, nonetheless interplanetary limits necessitate a transitory conversation of consequences here.

HDFS on linux agonizes after the alike presentation glitches as on FreeBSD, while the grade varies by filesystem and test. Simultaneous writes on Linux exhibited healthier presentation physiognomies than FreeBSD. For example, the ext4 filesystem presented an 8% squalor touching among 1 and 4 simultaneous writers, while the XFS filesystem presented no squalor in the alike test. This likens to a 47% droplet in FreeBSD as exposed in figure 5(a). In contrast, HDFS on linux consumed worse presentation for simultaneous reads than FreeBSD. the ext4 filesystem tainted by 42% touching after 1 to 4 simultaneous readers, and XFS tainted by 43%, likened to 21% on freebsd as exposed in figure 5(b). Finally, disintegration was abridged on Linux, as the ext4 filesystem tainted by 8% and the XFS filesystem by 6% when a single booklover protected annals shaped by 1 to 4 simultaneous writers. This likens to a 42% squalor in FreeBSD, as exposed in figure 6.

Hadoop in Windows 7 trusts on a UNIX emulation layer such as Cygwin to function. Write Band-width to diskette was acceptable (approximately 60MB/s), nonetheless read Band-width was very low (10MB/s or less) notwithstanding tall diskette utilization (in additional of 90%). while the aim of this presentation squalor was not investigated closely, the conduct is steady with diskette admittance designs using minor i/o needs (2kb-4kB) in its home of big needs (64kB and up). since of these presentation limitations, Hadoop in Windows is used only for nonperformance-critical appeal development. all large-scale placements of Hadoop in manufacturing use Unix-like working schemes such as freebsd or Linux, which are the emphasis of this paper.

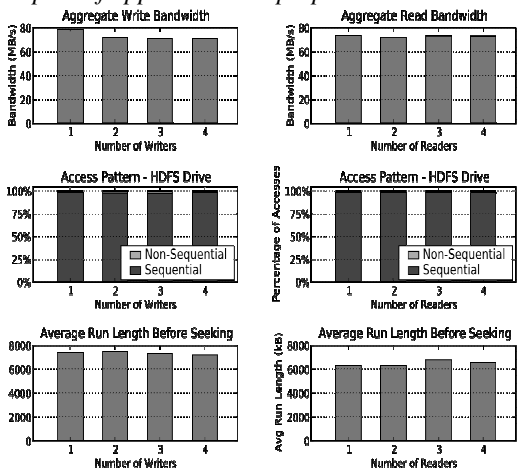
IV. PREPAR E YOUR PAPER BEFORE STYLING

As branded in unit III, the moveable application of HDFS agonizes after a amount of blockages shaped by inferior heights of the software stack. these glitches include:

Disk preparation — the presentation of simultaneous booklovers and writers agonizes after deprived diskette scheduling, as understood in unit III-E1. while HDFS clienteles admittance huge annals in a flowing fashion, the outline gulfs all folder hooked on manifold HDFS parts (typically 64MB) and lesser packs (64kB). the appeal watercourse essentially obtainable to the diskette is inserted among simultaneous clienteles at this minor granularity, compelling dangerous pursues and humiliating bandwidth, and negating one of the key possible welfares that a big 64MB part size would consume in enhancing simultaneous diskette accesses.

(a) Simultaneous Writers (b) simultaneous Readers

Fig. 7. Impact of appeal diskette preparation on simultaneous artificial Writers and Readers



Filesystem allocation — in addition to deprived I/O scheduling, HDFS also agonizes after folder disintegration when distribution a diskette among manifold writers. As deliberated in unit III-E2, the all-out likely folder contiguity — the size of an HDFS part — is not conserved by the general-purpose filesystem when diskette allocation choices are made.

Filesystem sheet accumulation above — handling a filesystem sheet accumulation executes a calculation and memory above on the host system, as deliberated in unit III-D. This above is unnecessary since the flowing admittance designs of mapreduce presentations consume insignificant locality that can be browbeaten by a cache. Further, smooth if an exact appeal did advantage after a cache, the sheet accumulation supplies figures at the wrong granularity (4-16kB sheets vs 64MB HDFS blocks), thus needful additional work to apportion memory and achieve metadata.

To recuperate the presentation of HDFS, there are a variety of architectural developments that forte be used. In this section, moveable answers are chief discussed, shadowed by no portable answers that forte recuperate presentation additional at the expenditure of compromising a key HDFS arrangement goal.

A. Appeal Diskette Scheduling

A moveable way to recuperate diskette preparation and filesystem allocation is to adapt the way HDFS batches and gifts storing needs to the working system. In the current

Hadoop implementation, clienteles exposed a new socket to the DataNode to admittance figures at the HDFS part level. The Data-node spawn's one thread per customer to achieve composed the diskette admittance and net communication. All lively threads admittance the diskette concurrently. In a new Hadoop application using application-level diskette scheduling, the HDFS Data-node was altered to use two collections of threads: a set to grip per-client communication, and a set to grip per-disk folder access. Customer threads attach with clienteles and line unresolved diskette requests. Diskette threads — all accountable for a single diskette — choice a storing appeal for a exact diskette after the queue. all diskette group thread has the ability to interleave needs after dissimilar clienteles at whatever granularity is essential to attain filled diskette Band-width — for example, 32MB or above as exposed in figure 2. In the new configuration, needs are explicitly inserted at the granularity of a 64MB HDFS block. After the viewpoint of the OS, the diskette is protected by a single client, avoiding any OS-level preparation problems. The preceding examinations were repeated to inspect presentation under manifold writers and readers. The consequences are exposed in figure 7(a) and figure 7(b).

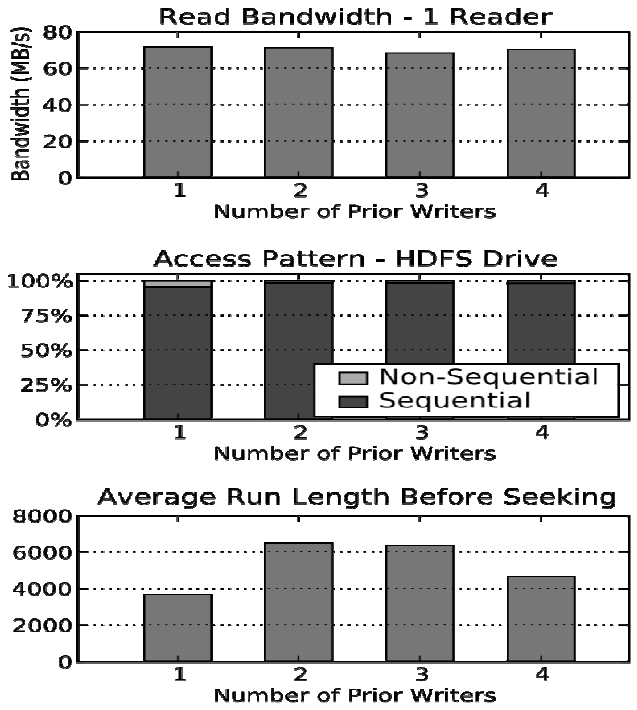


Fig. 8. Impact of appeal diskette preparation on figures Fragmentation

Compared to the preceding simultaneous writer consequences in figure 5(a), the better consequences exposed in figure 7(a) are striking. What was before a 38% presentation droplet when touching among 1 and 2 writers is currently an 8% decrease? Chance pursues consume remained closely totally eliminated, and the diskette is currently consistently protected in consecutive runs of better than 6MB. Simultaneous booklovers also show an alike development when likened against the preceding consequences in figure 5(b). In addition to refining presentation under simultaneous workloads, application-level diskette preparation also meaningfully abridged the quantity of figures disintegration created. Memory that, as exposed in figure 6, annals shaped with 2 simultaneous writers were split hooked on fragments of under 300kB. However, when retesting the alike trial with the adapted DataNode, the disintegration size exceeded 4MB, thus allowing abundant advanced diskette Band-width as exposed in figure 8.

Although this moveable development to the HDFS building better presentation significantly, it did not totally near the presentation gap. While the faultless consecutive run distance is in additional of 32MB, this alteration only attained run distance of about 6-8MB, notwithstanding presenting needs in abundant superior 64MB collections to the working scheme for service. To near this hole completely, non-portable methods are wanted to apportion big annals with better contiguity and less metadata.

B. Non-Portable Solutions

Some presentation blockages in HDFS, counting folder disintegration and accumulation overhead, are problematic to eradicate via moveable means. A amount of non-portable optimizations can be used if supplementary presentation is desired, such as bringing usage suggestions to the working system, choosing an exact filesystem for greatest performance, avoiding the filesystem sheet cache, or eliminating the filesystem altogether.

OS suggestions — Operating-system exact scheme noises can be used to decrease diskette disintegration and accumulation above by permitting the appeal to deliver "hints" to the underlying system. Certain filesystems allow annals to be pre-allocated on diskette without writing all the figures immediately. By apportioning storing in a single procedure in its home of frequent minor operations, folder contiguity can be greatly improved. as an example, the Data-node forte use the Linux-only fallocate() scheme noise in combination with the ext4 or XFS filesystems to pre-allocate interplanetary for an whole HDFS part when it is originally created, and advanced seal the unfilled area with appeal data. In addition, certain working schemes allow presentations to designate that sure sheets will not be reused after the diskette cache. Thus, the Data-node forte also use the posix fadvise scheme noise to deliver suggestions to the working scheme that figures protected will not be re-used, and henceforth accumulating must be a low priority. The third-party jposix java collection forte be used to allow this functionality in Hadoop, nonetheless only for exact stages such as linux 2.6 / AMD64.

Filesystem assortment — Hadoop placements forte mandate that HDFS be used only with local filesystems that deliver the wanted allocation properties. For example, filesystems such as XFS, ext4, and others provision extents of varying sizes to decrease folder disintegration and recuperate treatment of big files. While HDFS is written in a moveable manner, if the underlying filesystem behaves in such a fashion, presentation forte be meaningfully enhanced. Similarly, using a deprived local filesystem will damage HDFS.

Cache avoid — in linux and FreeBSD, the filesystem sheet accumulation can be avoided by opening a folder with the O straight flag. Folder figures will be straight transported via straight memory admittance (DMA) among the diskette and the user-space shield specified. This will avoid the accumulation for folder figures (but not filesystem metadata), thus eliminating the computer above consumed allocating, locking, and deallocating pages. While this can recuperate presentation in HDFS, the application is non-portable. Using dma transmissions to user-space needs that the appeal shield is associated to the expedient part size (typically 512 bytes), and such provision is not if by the java Virtual Machine. The java innate border (JNI) forte be used to tool this functionality as a minor innate routine (written in c or C++) that unbolts annals using o DIRECT. the innate cypher necessity achieve memory allocation (for alignment purposes) and deallocation later, as Java's innate garbage collection topographies do not spread to cypher appealed by the JNI.

Local filesystem Elimination — to exploit scheme performance, the HDFS Data-node forte avoid the os filesystem completely and straight achieve folder allocation on a raw diskette or partition, in spirit substituting the kernel-provided filesystem with a tradition application-level filesystem. A tradition filesystem forte decrease diskette disintegration and group above by apportioning interplanetary at a superior granularity (e.g. at the size of an HDFS block), permitting the diskette to function in an additional well-organized way as exposed in figure 2.

To count the best-case development likely with this technique, shoulder an idealized on-disk filesystem where only 1 diskette pursue is wanted to but all HDFS block. Since of the big HDFS part sizes, the quantity of metadata wanted is low and forte be cached in DRAM. in such a system, the even run distance beforehand looking for must be 64MB, likened with the 6MB runs got with application-level preparation on a conservative filesystem (See figure 7). on the test phase using an artificial diskette utility, cumulative the run distance after 6MB to 64MB recovers read Band-width by 16MB/s and write Band-width by 18MB/s, a 19% and 23% improvement, respectively. Using a less optimistic approximation of filesystem efficiency, smooth cumulative the run distance after 6MB to 16MB will recuperate read Band-width by 14 mb/s and write Band-width by 15 MB/s, a 13% and 19% improvement, respectively.

V. LINKED WORK

HDFS waiters (i.e., DataNodes) and outdated flowing television waiters are composed used to provision customer presentations that consume admittance designs branded by long consecutive reads and writes. As such, composed schemes are architected to errand tall storing Band-width over low admittance dormancy [20]. Outside this, however, there are key supplies that differentiate flowing television waiters after HDFS servers. First, flowing television waiters essential to degree pace to safeguard that the all-out amount of simultaneous clienteles obtains the wanted facility level. In contrast, mapreduce clienteles running batch-processing non-interactive presentations are dormancy insensitive, permitting the storing scheme to exploit general bandwidth, and thus bunch cost-efficiency. Second, television waiters frequently provision differentiated facility heights to dissimilar appeal streams, while in HDFS all clienteles consume equal priority. Occupied collectively, these supplies consume interested the arrangement of a big amount of diskette preparation procedures for television waiters [5], [7], [14], [19], [20], [21]. All procedure brands dissimilar tradeoffs in the goals of if scheduler fairness, meeting firm or soft facility deadlines, plummeting memory shield requirements, and minimalizing drive seeks.

In addition to resemblances with flowing television servers, HDFS waiters also portion resemblances with catalogues in that composed are used for data-intensive calculating presentations [18]. But, catalogues typically brand dissimilar arrangement selections that errand presentation in its home of portability. First, while Hadoop is written in java for portability, catalogues are typically written in low-level appeal languages to exploit performance. Second, while Hadoop only uses java innate folder i/o features, profitable catalogues deed OS-specific noises to enhance filesystem presentation for an exact phase by configuring or avoiding the kernel sheet cache, utilizing straight I/O, and working folder fastening at the node equal [10], [15]. Third, while HDFS trusts on the innate filesystem for portability, frequent well-known catalogues can be prearranged to straight achieve storing as raw disks at the appeal level, avoiding the filesystem completely [1], [12], [16]. Using storing in this way permits the filesystem sheet accumulation to be avoided in errand of an appeal cache, which eliminates double-buffering of data. Further, avoiding the filesystem delivers the appeal fine-grained switch over diskette preparation and allocation to decrease disintegration and seeks. Thus, catalogues show the presentation that can be augmented if movability is sacrificed or if supplementary application exertion is exerted to provision manifold stages in dissimilar manners.

One exact eye of catalogue arrangement — application level I/O preparation — feats appeal admittance designs to exploit storing Band-width in a way that is not likewise exploitable by HDFS. Application-level I/O preparation is frequently used to recuperate catalogue presentation by plummeting pursues in schemes with big figures of simultaneous queries. Since catalogue workloads frequently consume figures re-use (for example, on communal indexes), storing usage can be abridged by distribution figures among lively enquiries [6], [24]. Here, part or all of the diskette is unceasingly scanned in a consecutive manner. Clienteles join the image watercourse in-flight, permission after they consume conventional all essential figures (not unavoidably in-order), and not ever interrupt the watercourse by triggering instant seeks. In this way, the uppermost general throughput can be upheld for all queries. This exact type of preparation is only beneficial when manifold clienteles all admittance certain helping of communal data, which is not communal in frequent HDFS workloads.

Some optimizations future currently for Hadoop may be current in the google-developed mapreduce application that

is not openly available. the optimizations branded for the google application comprise plummeting diskette pursues for writes by batching and categorization central data, and plummeting diskette pursues for reads by keen preparation of needs [9].

VI. CONCLUSIONS

The presentation of MapReduce, and Hadoop in particular, has remained called hooked on query recently. For example, in certain experiments, presentations using Hadoop did ill likened to alike agendas using alike catalogues [18], [22]. While such changes are typically blamed on the mapreduce paradigm, this newspaper shows that the underlying filesystem can consume an important influence on the general presentation of a mapreduce framework. Enhancing HDFS as branded in this newspaper will development the general competence of mapreduce presentations in Hadoop. While this may or may not alteration the ultimate deductions of the mapreduce versus alike catalogue debate, it will surely allow a fairer judgment of the genuine software design models.

Furthermore, the presentation influences of HDFS are frequently concealed after the Hadoop user. While Hadoop delivers integral functionality to outline map and decrease task execution, there are no integral tools to outline the outline itself, permitting presentation blockages to continue hidden. This newspaper is the chief to tag the connections among Hadoop and storage. Here, we clarified how frequent presentation blockages are not straight attributable to appeal cypher (or the mapreduce software design style), nonetheless somewhat are shaped by the task scheduler and dispersed filesystem underlying all Hadoop applications.

The deprived presentation of HDFS can be attributed to examinations in upholding portability, counting diskette preparation under simultaneous workloads, filesystem allocation, and filesystem sheet accumulation overhead. HDFS presentation under simultaneous workloads can be meaningfully better through the use of application-level I/O preparation while preservative portability. Additional developments by plummeting disintegration and accumulation above are also possible, at the expenditure of plummeting portability. However, upholding Hadoop movability whenever likely will abridge development and advantage users by plummeting installation complexity, thus hopeful the spread of this alike calculating paradigm.

REFERENCES

[1] Chandrasekar, S. ; Dakshinamurthy, R. ; Seshakumar, P.G. ;Prabavathy, B. ; Babu, C. “A novel indexing scheme for efficient handling of small filesin Hadoop Distributed File System”Computer Communication and Informatics (ICCCI), 2013 International Conference on Publication Year: 2013, Page(s): 1 - 8

[2] Jing Zhang ; Gongqing Wu ; Xuegang Hu ; Xindong Wu “A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services” Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on Publication Year: 2012 , Page(s): 12 - 21

[3 Kaushik, R.T. ; Bhandarkar, M. ; Nahrstedt, K. “Evaluation and Analysis of GreenHDFS: A Self-Adaptive, Energy-Conserving Variant of the Hadoop Distributed File System” Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on Publication Year: 2010 , Page(s): 274 - 287

[4] Hieu Hanh Le ; Hikida, S. ; Yokota, H. Dependable, “An Evaluation of Power-Proportional Data Placement forHadoop Distributed File Systems Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on Publication Year: 2011 , Page(s): 752 - 759

[5] Attebury, G. ; Baranovski, A. ; Bloom, K. ; Bockelman, B. ; Kcira, D. ;Letts, J. ; Levshina, T. ; Lundestedt, C. ; Martin, T. ; Maier, W. ;Haifeng Pi ; Rana, A. ; Sfiligoi, I. ; Sim, A. ; Thomas, M. ;Wuerthwein, “Hadoop distributed file system for the Grid” F. Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE Publication Year: 2009 , Page(s): 1056 - 1061

[6] Youwei Wang ; WeiPing Wang ; Can Ma ; Dan Meng “Zput: A speedy data uploading approach for the Hadoop Distributed File System” Cluster Computing (CLUSTER), 2013 IEEE International Conference on Publication Year: 2013 , Page(s): 1 – 5

[7] Shahabinejad, M. ; Khabbazian, M. ; Ardakani, M. “An Efficient Binary Locally Repairable Code for Hadoop Distributed File System” Communications Letters, IEEE Volume: 18 , Issue: 8 Publication Year: 2014 , Page(s): 1287 – 1290

[8] Tomasic, I. ; Ugovsek, J. ; Rashkovska, A. ; Trobec, R. “Multicluster Hadoop Distributed File System”MIPRO, 2012 Proceedings of the 35th International Convention Publication Year: 2012 , Page(s): 301 - 305

[9] Hsiao-Ying Lin ; Shiuan-Tzuo Shen ; Wen-Guey Tzeng ; Lin, B.-S.P. “Toward Data Confidentiality via Integrating Hybrid Encryption Schemes and Hadoop Distributed File System” Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on Publication Year: 2012, Page(s): 740 - 747

[10] Moraveji, R. ; Taheri, J. ; Farahabady, M.R.H. ; Rizvandi, N.B. ;Zomaya, A.Y. “Data-Intensive Workload Consolidation for the Hadoop Distributed File System” Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on Publication Year: 2012 , Page(s): 95 – 103 “The Hadoop Distributed File System”

[11] Shvachko, K. ; Hairong Kuang ; Radia, S. ; Chansler, R. Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on Publication Year: 2010 , Page(s): 1 - 10

[12] Xiayu Hua ; Hao Wu ; Shangping Ren “Enhancing Throughput of Hadoop Distributed File System for Interaction-Intensive Task Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on Publication Year: 2014 , Page(s): 508 – 511

[13] Wei Zhou ; Jizhong Han ; Zhang Zhang ; Jiao Dai ,“Dynamic Random Access for Hadoop Distributed File System” Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on Publication Year: 2012 , Page(s): 17 – 22

[14] Madaan, S. ; Agrawal, R.K. “Implementation of identity based distributed cloud storage encryption scheme using PHP and C for Hadoop File System” Tier 2 Federation Grid, Cloud & High Performance Computing Science (RO-LCG), 2012 5th Romania Publication Year: 2012 , Page(s): 74 - 77

[15] Yang Jin ; Tang Deyu ; Zhou Yi “A Distributed Storage Model for EHR Based on HBase” Information Management,

Innovation Management and Industrial Engineering (ICIII), 2011 International Conference on Volume: 2 Publication Year: 2011 , Page(s): 369 - 372

[16] Yang Jin ; Tang Deyu ; Zheng Xianrong "Research on the distributed electronic medical records storage model" IT in Medicine and Education (ITME), 2011 International Symposium on Volume: 2 Publication Year: 2011 , Page(s): 288 - 292

[17] Krish, K.R. ; Anwar, A. ; Butt, A.R. "hatS: A Heterogeneity-Aware Tiered Storage for Hadoop" Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on Publication Year: 2014 , Page(s): 502 – 511

[18] Changjian Fu ; Zhihua Leng "A Framework for Recommender Systems in E-Commerce Based on Distributed Storage and Data-Mining" E-Business and E-Government (ICEE), 2010 International Conference on Publication Year: 2010 , Page(s): 3502 - 3505

[19] Bo Dong ; Xiao Zhong ; Qinghua Zheng ; Lirong Jian ; Jian Liu ; Jie Qiu ; Ying Li "Correlation Based File Prefetching Approach for Hadoop" Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on Publication Year: 2010 , Page(s): 41 - 48

[20] Bo Dong ; Jie Qiu ; Qinghua Zheng ; Xiao Zhong ; Jingwei Li ; Ying Li "A Novel Approach to Improving the Efficiency of Storing and Accessing Small Files on Hadoop: A Case Study by PowerPoint Files" Services Computing (SCC), 2010 IEEE International Conference on Publication Year: 2010 , Page(s): 65 - 72

[21] Martha, V.S. ; Weizhong Zhao ; Xiaowei Xu "h-MapReduce: A Framework for Workload Balancing in MapReduce" Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on Publication Year: 2013 , Page(s): 637 - 644

[22] Li Wang ; Zhiwei Ni ; Yiwen Zhang ; Zhang Jun Wu ; "Liyang Tang Notice of Violation of IEEE Publication Principles "Pipelined-MapReduce: An Improved MapReduce Parallel Programing Model " Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on Volume: 1 Publication Year: 2011 , Page(s): 871 - 874

[23] Kaiqi Xiong ; Yuxiong He "Power-effiicent resource allocation in MapReduce clusters"Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on Publication Year: 2013 , Page(s): 603 – 608

[24] Yanpei Chen ; Ganapathi, A. ; Griffith, R. ; Katz, R. "The Case for Evaluating MapReduce Performance Using Workload Suites" Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on Publication Year: 2011 , Page(s): 390 - 399 "Getting more for less in optimized MapReduce workflows"

[25] Zhuoyao Zhang ; Cherkasova, L. ; Boon Thau Loo Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on Publication Year: 2013 , Page(s): 93 – 100

[26] Guigang Zhang ; Jian Wang ; Weixing Huang ; Chao Li ; Yong Zhang; Chunxiao Xing "A Semantic++ MapReduce: A Preliminary Report"Semantic Computing (ICSC), 2014 IEEE International Conference on Publication Year: 2014 , Page(s): 330 – 336

[27] Kewen Wang ; Wenzhong Predator An experience guided configuration optimizer for Hadoop MapReduce Tang Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on Publication Year: 2012 , Page(s): 419 - 426

[28] Chu Huang ; Sencun Zhu ; Dinghao Wu Towards Trusted Services: Result Verification Schemes for MapReduce Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on Publication Year: 2012 , Page(s): 41 - 48

[29] Cura: A Cost-Optimized Model for MapReduce in a Cloud Palanisamy, B. ; Singh, A. ; Langston, B. Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on Publication Year: 2013 , Page(s): 1275 – 1286

[30] M2M: A simple Matlab-to-MapReduce translator for cloud computing Zhang, Junbo ; Xiang, Dong ; Li, Tianrui ; Pan, Yi Tsinghua Science and Technology Volume: 18 , Issue: 1 Publication Year: 2013 , Page(s): 1-9

