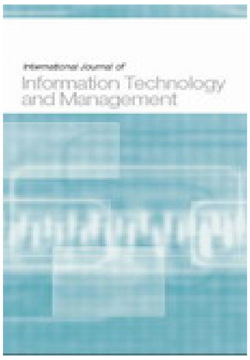


International Journal of Information Technology and Management

ISSN online: 1741-5179 - ISSN print: 1461-4111
https://www.inderscience.com/ijitm

Big data block impact within big data environment

Ron Ziv, Oded Koren, Nir Perel

DOI: 10.1504/IJITM.2023.10055156

Article History:

Received:	14 November 2017
Last revised:	29 October 2018
Accepted:	14 May 2019
Published online:	05 April 2023

Big data block impact within big data environment

Ron Ziv and Oded Koren

School of Industrial Engineering and Management,
Shenkar – Engineering. Design. Art.,
12 Anne Frank St., Ramat-Gan, Israel
Email: zivron@gmail.com
Email: odedkoren@shenkar.ac.il

Nir Perel*

School of Industrial Engineering and Management,
Afeka Tel-Aviv Academic College of Engineering,
38 Mivtza Kadesh St., Tel-Aviv, Israel
Email: perelnir@gmail.com
*Corresponding author

Abstract: Handling data is becoming more and more complex. A higher velocity of data is created as more people have access to data generating devices such as computers, mobile phones, medical devices, home appliances, etc. Data files, such as user activity logs, system logs and so on, are stored in HDFS™ big data platform in various sizes, which takes into consideration the business requirements, infrastructure parameters, administration decisions, and other factors. Dividing the data files (in various volumes) without taking into consideration the HDFS™ predefined block size, may create performance issues that can affect the system's activity. This paper presents how HDFS™ block design affects the performance of Apache™ Hadoop® big data environment by testing different architectures for reading, writing, and querying identical datasets. We designed three scenarios to illustrate different file divisions on the big data platform. The findings present a significant impact on the performance of a system in accordance with the architecture deployed.

Keywords: information management; architecture; HDFS; performance; big data; block partition.

Reference to this paper should be made as follows: Ziv, R., Koren, O. and Perel, N. (2023) 'Big data block impact within big data environment', *Int. J. Information Technology and Management*, Vol. 22, Nos. 1/2, pp.140–159.

Biographical notes: Ron Ziv received his MSc in Industrial Engineering and Management from Shenkar – Engineering. Design. Art. Currently, he is as an internal auditor, specialising in information systems at Teva Pharmaceutical Industries Ltd. He is also a Certified Information Systems Auditor (CISA), ISACA. His interests include strategic management, risk management, project management, configuration management, data management, business intelligences, information security and privacy, audit, and big data.

Oded Koren has a PhD from Tel-Aviv University, Israel. He is a full faculty member at the School of Industrial Engineering and Management in Shenkar – Engineering, Design, Art. His research interests are in big data and AI domains.

Nir Perel received his PhD in Operations Research from Tel-Aviv University, Israel. He is a Senior Lecturer at the School of Industrial Engineering and Management, Afeka Tel-Aviv Academic College of Engineering. His research interests include operations research modelling, queuing theory and statistical learning.

1 Introduction

1.1 Big data

Data is obtained by various means such as observation and measurements. The collection of data is used to describe attributes of items or something that has happened, is happening, or will happen (Ackoff, 1989). Today, we create about 2.5 Exabyte of data every day (IBM, 2017). Furthermore, 90% of the data which exists today was created only since 2015–2016 (IBM, 2017). Approximately 3.4 Exabyte of data (new and existing) is transferred over the internet each day. A research of global IP traffic indicates that by 2021 a 290% increase of global IP traffic will occur, resulting in approximately 9.9 Exabyte of data being transferred daily (Chandrasekar et al., 2013). The data created today is used for various purposes such as gaming, marketplace, social networking, communications, entertainment, etc. Data comes in the form of structured data from relational databases (rows and columns), semi-structured data (CSV, logs, XML, JSON), unstructured data (emails, documents, PDFs) and even binary data (images, audio, and video) (Li et al., 2008; Warden, 2011).

The term big data is subjective, and differs between individuals, organisations, and other types of entities. What constitutes 'big' needs to be placed in the context of the volume, velocity, and variety of the data (Beyer, 2011). To create value from all this data, one must take into consideration how data is generated, aggregated, analysed, and later consumed. To do these activities effectively and efficiently, data scaling needs to be incorporated as part of the design. There are generally two methods to tackle this challenge (Warden, 2011), namely vertical scaling and horizontal scaling vertical scaling can be achieved by increasing the processing speed of the computer simply by installing a faster processor, or by increasing the memory storage of the computer. This operation normally makes the management and control of the data easier as there are fewer computers needed for infrastructure, but also involves investing a large amount of resources in storage and processing hardware. Horizontal scaling allows for scaling by using a distributed system of lower cost computers which will create value from the data. This method can improve cost benefit, performance, and support of scalability of large amounts of datasets, but at the same time increases the complexity of data management and control such as in the case of fault tolerance, the quality of the data, data privacy and security – always an issue when data is distributed across a large number of machines (Katal et al., 2013). Furthermore, a pre-process on the data is the extract, transform, load (ETL) process. This process takes raw data, extracts the information required for analysis, transforms it into a proper format according to the business needs, and loads it to a data warehouse (Bansal, 2014).

1.2 Apache™[1] Hadoop®[2]

To deal with such a growing amount of data, several big data platforms were developed, both commercial and open source. This research focuses on the open source platform Apache™ Hadoop® which is a part of the Apache™ Software Foundation of open-source software projects [other big data storage technologies can be found in Siddiqa et al. (2017)].

Hadoop® was created by Doug Cutting, the developer of Apache™ Nutch™[3], an open-source web search engine (Warden, 2011). In 2006, Yahoo! Inc[4] continued to further develop the project when Doug Cutting joined the company (White, 2015).

Hadoop® is a framework that enables the storing and processing of large datasets in a distributed manner across multiple clusters of computers, i.e., a network of machines which are components of a larger system. This type of distributed file system is highly scalable in comparison to a traditional relational database management system (RDBMS), supporting Petabytes of data rather than Gigabytes (White, 2015).

Hadoop®'s framework core are the Hadoop® Common utilities, the Hadoop® Distributed File System (HDFS™)[5] (White, 2015), Hadoop® YARN[6] which is used for job scheduling and cluster resource management (White, 2015), and Hadoop® MapReduce[7] which is a system used for parallel processing of large datasets (Dean and Ghemawat, 2008; Papadimitriou and Sun, 2008) based on YARN (White, 2015). This HDFS™ infrastructure supports and enables to process large amounts of data coming from various sources (Storey and Song, 2017).

The HDFS™ core, which is based on Google File System (GFS) (see, Ghemawat et al., 2003), is responsible for the distribution of files across the system. Using parallel servers/computers (also known as datanodes on Hadoop platform) allows the user to store and analyse (via MapReduce) the data in the HDFS (Jach et al., 2015). To do so, it uses blocks (Wang et al., 2017; White, 2015) that divide the data files to parts (block sizes) within the datanodes with replications of the blocks. It is important to note that the common default in big data platform is three replications per each block (Reuther et al., 2018), which complicates the storage management and it is monitoring. According to Reuther et al. (2018), an inherent effect can be identified on an application's overall performance, caused by the number of read and write activities needed in relation to the amount of blocks allocated for these activities. The extent of influence on performance is demonstrated in the current paper. The predefined block size highly affects the number of divisions of files to be allocated within the datanodes (based on *HDFS*™). It is important to emphasise that the *HDFS*™ default block size are 64 MB or 128 MB as described in Nghiem and Figueira (2016). For example, in Warden (2011) the authors used 64 MB block size to examine replication loss in *HDFS*™.

The Hadoop® MapReduce core performs two critical tasks. The first is the map function which reads from the input files, and processes key/value data pairs to output intermediate files. The reduce function reads the new intermediate files and writes new records as the final output files, while performing any processing tasks assigned by the user and as identified as the key/values (Khan et al., 2014). In general, Hadoop® is based on *HDFS*™ for the data and on MapReduce for processing the data (Assunção et al., 2015). In addition to the core functions, application layer software components have been developed, such as Apache™ Hive™ (hereafter: Hive)[8]. Hive is a data warehouse infrastructure developed by Facebook©[9], which offers the ability to perform data summarisation, query, and analysis. Its scripting language, HiveQL is SQL-like. The queries are compiled into jobs executed on the Hadoop® platform (Thusoo et al., 2009).

Research studies have been performed on the benchmarking of different big data solutions and architectures. For example, an evaluation of Hadoop®'s HDFS™ and MapReduce has been tested alongside two database management systems (DBMS) using SQL, to understand the impact on performance and the complexity in system realisation (Pavlo et al., 2009). In Pavlo et al. (2009), the authors have revealed that parallel DBMS outperform Hadoop® by a factor of 3.2 for a DBMS-X platform, and 2.3 for a Vectra platform in comparison to the DBMS-X platform, however they have also noticed that Hadoop® is a preferred solution with regards to configuration setup time and the framework's flexibility for data types and user defined functions. The authors, while performing their study using 100 nodes, believed that the performance on 1,000 nodes would be similar.

An interesting problem is how to utilise HDFS when working with small files. In Chandrasekar et al. (2013) the authors studied the performance of HDFS when handling small files, while in Ahad and Biswas (2018) the authors suggested a method of merging small files according to their type and size. Furthermore, the issue of energy efficiency in Hadoop has been widely studied in Wu et al. (2018), where solutions for improving energy efficiency were suggested.

A further study performed analysis of datasets (Loebman et al., 2009). The authors loaded data files with sizes of 169 MB, 1.4 GB, and 36 GB, while comparing between a traditional RDBMS using parallel processing, and the Hadoop® framework loading data using PigLatin. Both environments were tested using a single node, 2-node, 4-node, and 8-node configurations. The authors concluded that the commercial RDBMS outperformed the Hadoop®, but also acknowledged that if hundreds or thousands of nodes are to be used, a parallel database is likely to fall short in performance.

Additional research was performed on the comparison between two Hadoop® framework-based solutions such as Pig™[10] and Hive™[11]. Both Pig™ and Hive™ are used for data processing over the Hadoop® platform (White, 2015). In Dhawan and Rathee (2013), the authors examined the scripting languages in context of their data flow, schema, and Turing completeness, by using map-reduce jobs. In both cases the researchers found that both scripting languages had similar performance results, but differed in their method of reaching desired results, and language completeness.

In Kendal et al. (2016), the authors compared between Pig™ and Hive™, while testing performance of 1 GB and 2 GB datasets on a single node. The conclusion was that Hive™ is more efficient, as fewer actions were needed to produce the same results. The authors also concluded that Hive™ will be more suitable for large files aggregation.

In Stewart et al. (2011), three high level query languages were examined: HiveQL, PigLatin, and JAQL. The authors concluded that HiveQL presents the fastest results and that both HiveQL and PigLatin are simplified scripting languages with respect to the number of code lines needed to execute operations.

An additional research (Engelberg et al., 2016) also focused on Pig™ and compared different decentralisation levels, i.e., cluster sizes, of a single node, 3 nodes, and 22 nodes, by manipulating a file of 1.9 GB. The research indicated that processing times improved with the introduction of additional nodes into the overall architectureup to a certain threshold.

In Andreolini et al. (2015) an adaptive algorithm to improve monitoring of big data applications was examined, while in Wang et al. (2017) the data replication (e.g., block replication) between datanodes was investigated.

Despite all the studies mentioned above, there is currently a lack of literature on the effect of different system architectures and how the utilisation of various amounts of blocks available within a big data system affects the system's performance. Specifically, the case where the block size is predefined by the system's default settings, and is not configured based on the organisation's real business needs and requirements. Such differences in architecture and block utilisation may result in substantial performance issues for a system, and the resources an organisation might need to invest to achieve desired outputs. To measure the influence on performance statistically, it is necessary to disassemble the procedures into the various jobs executed by MapReduce, and later to analyse the individual tasks performed, rather than measuring the overall elapsed time. By doing so, it is possible to identify which section of the procedure is affected by utilisation blocks and which section is indifferent.

2 Research goal

The goal of this research is to provide statistical and objective data which in return could be leveraged by enterprises and individuals alike in the design, development, and deployment of Hadoop®-based file distribution and by the use of Hive™ for the processing of big data. Within this research we focus on the effect of the number of blocks used to ingest the exact same amount of data, and the influence of different architectures, on the overall system performance. To achieve this goal, the following objectives were defined:

- Design and execute multiple pre-planned scenarios to be tested in order to identify the performance sensitivity between different architectures, and to make results available for business and/or research applications.
- Document audit logs used for the analysis of the individual jobs which were processed to support research results.

3 Research method

3.1 Case study's data

To support the study, we created randomised numerical and textual data. The data was stored in a 360 MB file containing 584,834 records used for scenario A. Next, we copied the original file (360 MB) and split it into two 180 MB files, containing 292,417 records each. These files were used as part of scenario B. Lastly, we copied again the original file (360 MB) and split it into three 120 MB files, containing 194,945 records in the first two files and 194,944 records in the third file. These last three files were used for scenario C.

In the next step, we replicated each of the files 20 times in accordance to the predefined number of test runs planned to be executed per scenario. All files contain the exact same data structure, i.e., the assigned attributes naming and data types are similar (num1, num2, num3, num4, num5, num6, text1, text2).

Figure 1 Example of records ingested

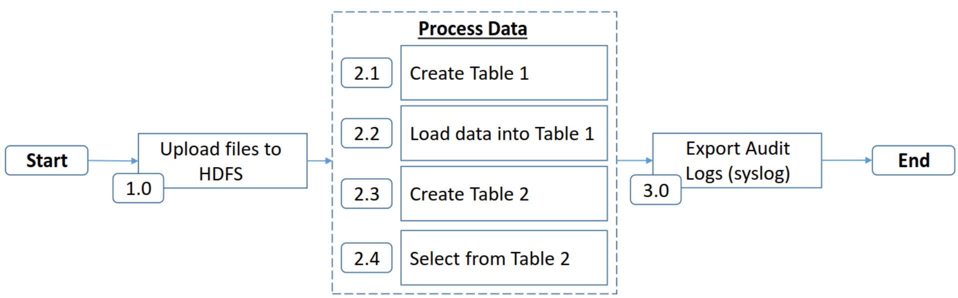
```
51697204,43785492,26668849,89878477,95923033,90456285,mqak kw aknz hkpi aloj ge nvkk vhwb kuej ow gbob ot
33857868,80189043,15842482,73918885,71065117,81223038,xjdj ak frep pspb xzuc dw hifx rgus ikdp yd arge lw
17539298,31239989,89931470,50226176,98975789,82783087,fwvy em lvgt ngza qvrt do wqbu slue yvpk oj fzse gn
20533525,91580206,67349002,93101287,694935,53444706,pbjh kq qgys jiwd kavt fi cgpd aiao ntjo sb xfov gmal
51835885,55679443,26704408,30596361,65760692,76616397,vqyc fz wmvu xgxa jftd oh xcry lici anxk ah usjx wp
75151119,75123235,90301005,65274761,56723390,39680129,gzwi hf gxxp onpt pgct zk kmhp zyrm fsgq pb gxge yj
15590928,29204491,55164884,35207770,92444200,81029024,xxfh bw hmfl dcay mneq ca tcez hlen njok ix kcff gp
63863700,10487740,32521631,90019440,43356100,24659434,fvqa zv rsxn wqdf uxrh no ozhi ztjm pfee nx tpzg xz
83174929,35761949,43225371,84061242,59286590,28345938,gohr og neey gtlp vdqb ph guap irec lzlm yi cwfi up
68326958,69228060,73014014,7811688,25422115,35698191,mwxh wc iplz cauk ifim bs hrpq ophe avpa st znue prv
75120081,86762088,7118950,55974064,79314884,82442605,gurd ok eydx odsa fqvp dv yekq tvju igdw gc xdmc qha
49810217,37736675,94620518,73341575,411559,63527166,oakq uo bdll cgic mhze hu ikwi drxt ukls zw boop frej
29998054,15429462,83012948,29062471,66371431,84149076,yesw ph ncfr havg nfud yx pigq hbsb chkr qd nwkl wl
6246303,82320682,83567140,2217267,37459338,26748989,nsck dw pgbh ihdh oslr va ixgj nlbr axad te cpwq pjoj
83308728,93250062,50544203,50265427,14491750,80840776,lmud lp wtdq lmps folf ri iyjf dhub mthq cg uvsy mc
65000759,3870543,94006072,82954171,53304585,92417881,upyd rh evtf oahs cmog sr chms bxdm baur di fqfv tii
81476683,5026768,12823532,86556616,70362115,54623592,gdev ly yvum rbko yzdf sf kppl gvyp rham qu eenp gcn
98289002,9544368,33791411,5513510,76433523,27414132,bgza dj kiwb zdok xbsu cu tkvs yuwl alnk ea usmz xxle
50187824,44491864,93932145,2979318,77691999,40398202,meaj qy rnwx ptwt osgj sd ewwm ltll kwit jm orjv dxx
11310706,48206403,6626001,26334144,40395255,95508226,yxwf pn jxyb cddg fsii kf srsf rgjm qsiw rh ybym rzu
30816567,81142792,86240242,37541574,20454084,42466610,kcay oo noxj dvft yyyj xv vvma dihq dyne qj kxfj gy
60735846,80856224,33051720,8098605,51318438,82467225,oiof oj ntcz vzww tisz ol ecyg kfvs qcde wx foxp tor
25225740,1445011,36182221,62828055,58612210,855015,xenx vc skma szvn yjlo vh scep okhq vgyb jd furp zyug
11133193,30956151,5300017,2404057,31884428,91053106,mwkh zr gkch bael zprg oa palp rwwy vida op wvgo xwdd
57297991,75867536,2382488,17082758,20680192,82103419,bile wc qtnb npuu zicm kp plqb ixxi riew sn lqya lrm
```

Figure 1 illustrates a representation of some records used within the study to support the various processes which shall be detailed in Subsection 3.2.

3.2 Case study data flow

As part of the research design process, we created a data flow to identify the various tasks to be processed, either manually or automated. Figure 2 visualises the data flow intended to be processed.

Figure 2 Data flow (see online version for colours)



Step 1.0 Upload files to HDFS™.

In this step, we uploaded manually into the HDFS™ a total of 120 files (e.g., 20 files for scenario A, 40 files for scenario B, and 60 files for scenario C). These files are the input data for step 2.0 below.

Step 2.0 Process data.

In this step, we used Hive™to process the data. Steps 2.1 through 2.4 can also be described in pseudocode syntax as follows:

```
[2.1]: Create TABLE 1
        num1:DOUBLE
        num2:DOUBLE
```

num3:DOUBLE

num4:DOUBLE

num5:DOUBLE

num6:DOUBLE

text1:STRING

text2:STRING

[2.2]: Load into TABLE 1 from files in HDFS™

[2.3]: Create TABLE 2, Attributes:

text1:STRING (from Table 1)

num7:DOUBLE (Count how many times the substring ‘ fm ‘ exists within the string)

text3:STRING (replace substring ‘ fm ‘ with a different substring ‘ AA ‘)

text2:STRING (from Table 1)

num9:DOUBLE (Count how many times the substring ‘ fm ‘ exists within the string)

text4:STRING(replace substring ‘ fm ‘ with a different substring ‘ AA ‘)

[2.4] Select * From TABLE 2 (to display the entire table)

Table 1 (input data representation) presents the input data which was loaded into the TABLE1 from the HDFS™, in accordance to process sequence numbers 2.1 and 2.2 as shown in Figure 2.

Table 2 (output data representation) presents the output data which was created in Table 2 of the database as part of process sequence numbers 2.3 and is the output as part of sequence 2.4, again as shown in Figure 2.

Step 3.0 Export audit logs,

In the last step, we exported all audit logs which documented the automated process 2.0 and saved them locally as standard text files (.txt) to be used for statistical analysis, further detailed in Section 4.

3.3 Case study scenarios

To meet our research objectives, we designed three scenarios. The scenarios differ by:

1. infrastructure resources used, i.e., the amount of blocks allocated by the HDFS™ per scenario
2. the amount of files ingested (1 × 360 MB, 2 × 180 MB, and 3 × 120 MB).

In addition, each scenario was tested 20 times to ensure reliability of the results.

Table 1 Input data representation (see online version for colours)

num1	*num2*	*num3*	*num4*	*num5*	*num6*	*text1*	*text2*
19210150	97006122	67266513	55737043	90163465	44515085	qhxr **fm** hgzo xbue velq jk wfzs rbcq vasx tm seno ebco ivoe mc bfer mmej yaad pg axut jkpt xcgz mi pigr dauc kalk it fogr qutb assr cq ihwo pczk jymf ss gemc jnkg aslh zg apij ccxi dide mf aqto lhre jrfy nr eqgf qrxz asjd bc kcwa amcn ijfv ik ubtl xyod icsu jg lmse hknr lemr aq yqta ghsa	rime ba bisk lvym oxqf kq cqzm oais haay ct riad cwka bpwf rc fgzb hpqe umsq xo vzav nkwy leyy hs qrib rbad qvgv st dlme ezvb zcbc qy eygg fsxx wqvp bb lcrx btaa yjmk rx gwph dhsd oaab dg qygf hvst vzat ly djfp lmrl cfqs fr goyv elbe vyzl ly eaef qtqr csrp tg kicy eyiq tpry xq aogi oqne
4941164	79552341	26821789	48947931	87745866	39583229	gtmn sz ucmt hgux auxl fe jiud dlux vgrp bz dhlm xeds suht sb bupp wqbn ageg jh hxti azci lxdp sp kcos cwuj exbs kx zswm gwma joae bk rnvk nbem fmpb ow dubu wmnd zuxm xd wuoy jpov tyws rd zzsm puly zowx vi eexw xbmd jfaq hw fzoi ptit lkdt pc vbch mebv gwiq mk kwmo frax tjpl tr vnxr dqur	rjae ga citv ltyb vxcs or gvpe bdwo fzzd zg fedr ogtp fwak oy idei fkvw rpui bn dwhh mgkq scdi dc iwyc chgy cnpj in pjao wmjw osnt uv mrbu dgjy drpj ax qiwc cikv osko gz seqm mhbi daey td yxqs tdzy xpbd ab iqwr frlh jhnx fw xvui ebhh pbhe bn whid grit szne yu uobp ismb kozo ha unvs rqpx
47426324	1155276	92737168	93745364	54154158	98271877	tjjc xc knmq lvya ziaq jt sqab rlok lxcr ze wywe ofti pghm kw gswu kmzo ohja fp dykv kamd adsl rx dxyg hwil xcey js coja uift ezjs ea rskq gngk ilpj mg ywvl pbvv bfoc uj epnm gieu jocy fc quni qbsy dkmn rc hpxw zcqo sqhg hd fbua druw jgdf gr fnar rxvj cwdl pn odyf cbby lprc da rnsj mohp	xpha xd wquu wdxf rgvy no fqis rrcx tpex ij jsyf iate wwfc **fm** owba lcgi vyza uq orbn ghnk zmrr rb zxvz tjtt xpwl rq gnom rony flop mp bzfs dqvw uigd il ahor kfwp rxpi bv jiiz shst vkoa **fm** tcxa scwt wtlx rj oqhk ryid hjkq **fm** ijal xnka jfsc rv kyqs cgsh yopb od ogpj mhfw naca **fm** obms fgws
64091056	78537719	85977282	36812030	49984197	15488532	gtoi **fm** pqet jndp kxrg **fm** xqet jtgr mxbi **fm** qndp gxqh fbnj pz zuvp okkn zlyy mu aabv ceoz brpu hw vsrc paqp npea hl atrx rkdr cuxb yr mrks fcgn dslu fg wdzz dffp epgl wa bwoz ymem bmip la ujll uvrx xbsi zr pnth aant nvfq ku izws bntz aluy db mema lbmg sgup mm qeno qejg ycsz va hdcy pzud	yyzz up dlqs djbi fojp fb mrdu vvuv osak jf famz bxvm yore uu mljp uoee nppp eb pefx qlfh tcqm vq vwbj wwze ahjo sq gxxn ilsb aaiy pb yfsk orrh szjc my effk dybq baxc au zskw qhbf essg yv hwqq vuwp arts pm uyre rais gekj up pknx zklo eqqn pn kjqq zibg gtun au yysu jxqh terc jf hytd uzmq
83748518	18277767	42436370	50918241	7636560	77551443	vxlo yo sutr cwdo pidi ls zxcb czop tyvm jk xdva hbrf qawh ze yeck tyry fyok ud baer hocb geix zp hunl daof eovq ll jgyl twgd gxqo gv jjae rdxv xuvs gi waxz yebn fjnl wt sixl hjtb nllr nk sizq qkhf xvdk zz bwxh cjxk fkvx it fwxt udys andy tb kdzb xewe evoc ji stlq sdne qgob wv pvmf bbhk	ilwn uk bodp lrdv jyhc kw flvw oerr sgve nl rvwd zgws zkoy kf gtwm zwep toli oe dywn mjjq arar cz csfh avaf jhkw mq btat rvbn kbjz mu qnqq hclb psur pt aazp kyrq corx up wovm fngc azin xc gkev tghd irmm qx snrg orsv ounl ka qitd bhmh lbbm qa yhvq tcrd qybk kb nfap kypa nurl **fm** sbzo qchf

Table 2 Output data representation (see online version for colours)

text1	*num7*	*text3*	*text2*	*num9*	*text4*
qhxr fm hgzo xbue velq jk wfzs rbcq vasx tm seno ebco ivoe mc bfer mmej yaad pg axut jkpt xcgz mi pigr dauc kalk it fogr qutb assr cq ihwo pczk jymf ss gemc jnkg aslh zg apij ccxi dide mf aqto lhre jrfy nr eqgf qrxz asjd bc kcwa amcn ijfv ik ubtl xyod icsu jg lmse hknr lemr aq yqta ghsa	1	qhxr AA hgzo xbue velq jk wfzs rbcq vasx tm seno ebco ivoe mc bfer mmej yaad pg axut jkpt xcgz mi pigr dauc kalk it fogr qutb assr cq ihwo pczk jymf ss gemc jnkg aslh zg apij ccxi dide mf aqto lhre jrfy nr eqgf qrxz asjd bc kcwa amcn ijfv ik ubtl xyod icsu jg lmse hknr lemr aq yqta ghsa	rime ba bisk lvym oxqf kq cqzm oais haay ct riad cwka bpwf rc fgzb hpqe umsq xo vzav nkwy leyy hs qrib rbad qvgv st dlme ezvb zcbc qy eygg fsxx wqvp bb lcrx btaa yjmk rx gwph dhsd oaab dg qygf hvst vzat ly djfp lmrl cfqs fr goyv elbe vyzl ly eaef qtqr csrp tg kicy eyiq tpry xq aogi oqne	0	
gtmn sz ucmt hgux auxl fe jiud dlux vgrp bz dhlm xeds suht sb bupp wqbn ageg jh hxti azci lxdp sp kcos cwuj exbs kx zswm gwma joae bk rnvk nbem fmpb ow dubu wmnd zuxm xd wuoy jpov tyws rd zzsm puly zowx vi eexw xbmd jfaq hw fzoi ptit lkdt pc vbch mebv gwiq mk kwmo frax tjpl tr vnxr dqur	0		rjae ga citv ltyb vxcs or gvpe bdwo fzzd zg fedr ogtp fwak oy idei fkvw rpui bn dwhh mgkq scdi dc iwyc chgy cnpj in pjao wmjw osnt uv mrbu dgjy drpj ax qiwc cikv osko gz seqm mhbi daey td yxqs tdzy xpbd ab iqwr frlh jhnx fw xvui ebhh pbhe bn whid grit szne yu uobp ismb kozo ha unvs rqpx	0	
tjjc xc knmq lvya ziaq jt sqab rlok lxcr ze wywe ofti pghm kw gswu kmzo ohja fp dykv kamd adsl rx dxyg hwil xcey js coja uift ezjs ea rskq gngk ilpj mg ywvl pbvv bfoc uj epnm gieu jocy fc quni qbsy dkmn rc hpxw zcqo sqhg hd fbua druw jgdf gr fnar rxvj cwdl pn odyf cbby lprc da rnsj mohp	0		xpha xd wquu wdxf rgvy no fqis rrcx tpex ij jsyf iate wwfc fm owba lcgi vyza uq orbn ghnk zmrr rb zxvz tjtt xpwl rq gnom rony flop mp bzfs dqvw uigd il ahor kfwp rxpi bv jiiz shst vkoa fm tcxa scwt wtlx rj oqhk ryid hjkq fm ijal xnka jfsc rv kyqs cgsh yopb od ogpj mhfw naca fm obms fgws	4	xpha xd wquu wdxf rgvy no fqis rrcx tpex ij jsyf iate wwfc AA owba lcgi vyza uq orbn ghnk zmrr rb zxvz tjtt xpwl rq gnom rony flop mp bzfs dqvw uigd il ahor kfwp rxpi bv jiiz shst vkoa AA tcxa scwt wtlx rj oqhk ryid hjkq AA ijal xnka jfsc rv kyqs cgsh yopb od ogpj mhfw naca AA obms fgws
gtoi fm pqet jndp kxrg fm xqet jtgr mxbi fm qndp gxqh fbnj pz zuvp okkn zlyy mu aabv ceoz brpu hw vsrc paqp npea hl atrx rkdr cuxb yr mrks fcgn dslu fg wdzz dffp epgl wa bwoz ymem bmip la ujll uvrx xbsi zr pnth aant nvfq ku izws bntz aluy db mema lbmg sgup mm qeno qejg ycsz va hdcy pzud	3	gtoi AA pqet jndp kxrg AA xqet jtgr mxbi AA qndp gxqh fbnj pz zuvp okkn zlyy mu aabv ceoz brpu hw vsrc paqp npea hl atrx rkdr cuxb yr mrks fcgn dslu fg wdzz dffp epgl wa bwoz ymem bmip la ujll uvrx xbsi zr pnth aant nvfq ku izws bntz aluy db mema lbmg sgup mm qeno qejg ycsz va hdcy pzud	yyzz up dlqs djbi fojp fb mrdu vvuv osak jf famz bxvm yore uu mljp uoee nppp eb pefx qlfh tcqm vq vwbj wwze ahjo sq gxxn ilsb aaiy pb yfsk orrh szjc my effk dybq baxc au zskw qhbf essg yv hwqq vuwp arts pm uyre rais gekj up pknx zklo eqqn pn kjqq zibg gtun au yysu jxqh terc jf hytd uzmq	0	
vxlo yo sutr cwdo pidi ls zxcb czop tyvm jk xdva hbrf qawh ze yeck tyry fyok ud baer hocb geix zp hunl daof eovq ll jgyl twgd gxqo gv jjae rdxv xuvs gi waxz yebn fjnl wt sixl hjtb nllr nk sizq qkhf xvdk zz bwxh cjxk fkvx it fwxt udys andy tb kdzb xewe evoc ji stlq sdne qgob wv pvmf bbhk	0		ilwn uk bodp lrdv jyhc kw flvw oerr sgve nl rvwd zgws zkoy kf gtwm zwep toli oe dywn mjjq arar cz csfh avaf jhkw mq btat rvbn kbjz mu qnqq hclb psur pt aazp kyrq corx up wovm fngc azin xc gkev tghd irmm qx snrg orsv ounl ka qitd bhmh lbbm qa yhvq tcrd qybk kb nfap kypa nurl fm sbzo qchf	1	ilwn uk bodp lrdv jyhc kw flvw oerr sgve nl rvwd zgws zkoy kf gtwm zwep toli oe dywn mjjq arar cz csfh avaf jhkw mq btat rvbn kbjz mu qnqq hclb psur pt aazp kyrq corx up wovm fngc azin xc gkev tghd irmm qx snrg orsv ounl ka qitd bhmh lbbm qa yhvq tcrd qybk kb nfap kypa nurl AA sbzo qchf

In accordance with our research goal, we control the amount of blocks allocated by HDFS™ per scenario. To do so, we configured HDFS™ to a 128 MB block allocation limit. By controlling block size limit, we ensure that for scenario A (1 file × 360 MB) three blocks will be used, for scenario B (2 files × 180 MB) 4 blocks will be used, and for scenario C (3 files × 120 MB) three blocks will be used. Table 3 presents the expected number of blocks HDFS™ will allocate per scenario as detailed above.

Table 3 Blocks allocation per scenario

Scenario	*# of files*	*Dataset volume (MB)*	*# of blocks allocated*
A	1	360	3
B	2	180	4
C	3	120	3

Figure 3 The block challenge (see online version for colours)

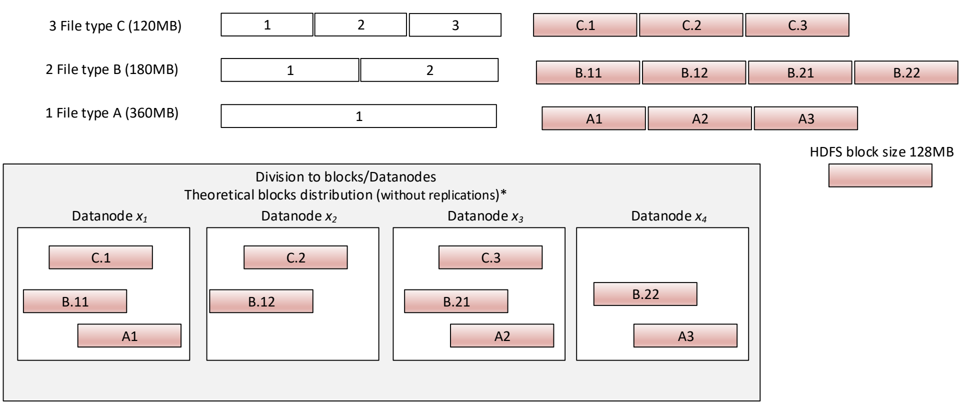


Figure 3 presents the challenge of predefined block deviation on different files. It is important to note that in real practice, many different files can have the same system logic/data/values, for example: saving user activities in logs, saving the systems process in logs, capturing network activities in logs, etc.

The issue examined in this paper arises in many cases, in which the system logs/data are divided according to different parameters, based on system administration procedures, infrastructure needs, system rules, business needs and logic. For example: data of the network activities can be captured and stored (per each file) according to date; system activities can be stored per predefined maximum file size, and so on.

The logic of storing the data in many files, without taking into consideration the HDFS™ predefined block size, may create performance issues of the system and other overheads. Figure 3 illustrate the files and block division that was selected for this research. As described above, all three file types (A, B and C) contain the same identical data, but in thee different scenarios. Figure 3 also presents the theoretical block distribution between datanodes. Note that the blocks may be distributed to different datanodes, as well as block replications (three replications per each block). This may increase the complexity of the system.

Table 4 Descriptive statistics for job 1 running times

	N	*Mean*	*Std. deviation*	*95% confidence interval for mean*		*Minimum*	*Maximum*
				Lower bound	*Upper bound*		
A	20	00:00:26.11	00:00:05.043	00:00:23.750	00:00:28.471	00:00:21.294	00:00:38.755
B	20	00:00:34.82	00:00:10.573	00:00:29.873	00:00:39.771	00:00:22.702	00:01:00.854
C	20	00:00:28.64	00:00:05.415	00:00:26.105	00:00:31.174	00:00:18.280	00:00:37.676
Total	60	00:00:29.86	00:00:08.201	00:00:27.739	00:00:31.976	00:00:18.280	00:01:00.854

The tests which were planned and executed can be found in Table 4, with each test identified by a unique identifier (test #). Table 5 (see Appendix A2) then displays the results which were observed in each test performed, identified by their assigned unique identification number.

Upon extraction and preliminary analysis of the test results based on the audit logs, it appeared that the MapReduce core function had separated the Hive™ source code into two jobs. The first job performed process sequence 2.1, 2.2, and 2.3 for the creation of Table 1 in the database, loading the data into Table 1, and creating Table 2 in the database respectively. The second job which was created was for the select function within process sequence 2.4, designed to display process results as outputs to the user, while providing assurance to the research team that the script developed functioned as intended.

In accordance with the observations mentioned above, we decided to focus the analysis efforts on job 1 running times. Job 1 inputs are different, based upon the particular scenario, and its final product (output) is a single table (see sequence 2.3 in Figure 3). Meanwhile, job 2 will always process the same data (job 1 output), thus job 2 running times are indifferent to the chosen scenario.

4 Research results

Our data consists of the 60 running times, divided into three groups according to the three scenarios as described in Subsection 3.3 above.

We study the running times of job 1, and the differences between these running times when examining the three scenarios (A, B and C). During the study it was evident that job 2 was not affected by the amount of blocks allocated by the system.

Figure 4 Boxplots for job 1 running times (see online version for colours)

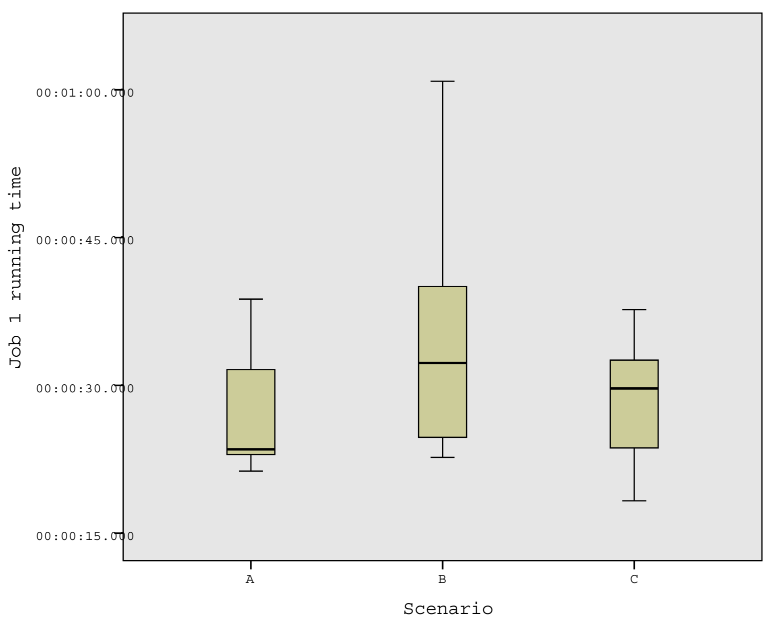
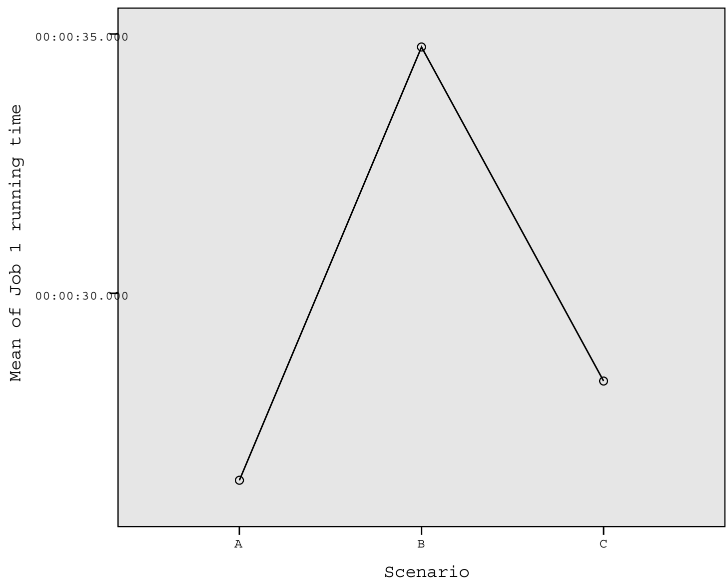


Figure 5 Means plot for job 1 running times



First, we present some descriptive statistics of the running times (for job 1) within the different scenarios. The outcomes are given in Table 4. Note that, as expected, the mean running times of scenarios A and C are smaller than the one in scenario B. Also, the standard deviation of the running time in scenario B is greater than in scenarios A and C. This occurs as a result of the data being divided into four blocks in scenario B, as opposed to three blocks in scenarios A and C. Figures 4 and 5 depict a boxplot and a means-plot for job 1 running times, divided by the three scenarios, respectively.

In order to statistically check differences between the running times (of job 1) in the three scenarios, we perform an analysis of variation (ANOVA) test. The results of the normality test show that the running times of job 1 in scenarios B and C are normally distributed, while in scenario A this is not the case. In addition, it is statistically shown that the variances of the running times are not equal within each scenario. Therefore, since some of the ANOVA assumptions are violated, we also performed the (nonparametric) Kruskal-Wallis test.

Table 5 ANOVA table

ANOVA					
Job 1 running time					
	Sum of squares	*df*	*Mean square*	*F*	*Sig.*
Between groups	803.435	2	401.718	7.236	0.002
Within groups	3,164.643	57	55.520		
Total	3,968.078	59			

Table 5 presents the ANOVA output. We observe a significance of 0.002, implying that the means of the running times in the three scenarios are not all equal to each other. In order to further investigate our hypothesis, we present in Table 6 a post-hoc (multi-comparisons) analysis via Fisher's least significant difference (LSD) test. As expected, we see significant differences between scenarios A and B, and between scenarios B and C.

Table 6 Results of Fisher's LSD multiple comparisons test

Multiple comparisons						
Job 1 running time *LSD*						
(I) scenario	*(J) scenario*	*Mean difference (I-J)*	*Std. error*	*Sig.*	*95% Confidence Interval*	
					Lower bound	*Upper bound*
A	B	00:00:08.71165	00:00:02.3562	0.000	–00:00:13.430	–00:00:03.993
	C	00:00:02.52880	00:00:02.3562	0.288	–00:00:07.247	00:00:02.190
B	A	00:00:08.71165	00:00:02.3562	0.000	00:00:03.993	00:00:13.430
	C	00:00:06.18285	00:00:02.3562	0.011	00:00:01.465	00:00:10.901
C	A	00:00:02.52880	00:00:02.3562	0.288	–00:00:02.190	00:00:07.247
	B	00:00:06.18285	00:00:02.3562	0.011	–00:00:10.901	–00:00:01.465

As mentioned before, since not all of the ANOVA assumptions were satisfied, we also performed a nonparametric test, Kruskal-Wallis test. The results, presented in Tables 7 and 8, imply, with a significance value of 0.003, that there is a difference between the distributions of the running times categorised by the three scenarios.

Table 7 Ranking results of Kruskal-Wallis test

Ranks			
	Scenario	*N*	*Mean rank*
Job 1 running time	A	20	21.15
	B	20	39.80
	C	20	30.55
	Total	60	

Table 8 Kruskal-Wallis test statistics

Test statistics[a,b]	
	Job 1 running time
Chi-square	11.405
Df	2
Asymp. sig.	0.003

Notes: [a]Kruskal-Wallis test.
[b]Grouping variable: scenario.

5 Conclusions and discussion

Every organisation has resources limitations. Whether it is an academic institution, a small business, a governmental agency, or a large global corporation, resources will always have limits. To reduce investments in computer hardware needed to process data, organisations need to architect their IT environments in an optimal manner, thus, making better use of resources and/or increasing profit. This study presents the effect on performance by use of different architectures within a big data environment.

We compared the performance of loading the exact same amount of data but with different volumes and number of files used, while controlling the environment to allocate either 3 or 4 blocks of 128 MB each. We processed the data by use of the same method, resulting in an identical output.

This analysis illustrates the issues which may arise when an organisation divides its data in a big data platform to a variety of file sizes, based on different parameters (time requirements, business logic, infrastructure issues, etc.), without taking into consideration the effects of the predefined HDFS™ block size.

Upon analysis of the results we identified using ANOVA a significant difference between scenarios using 3 or 4 blocks. However, since some of the ANOVA assumptions were violated, we continued by using the Kruskal-Wallis model (a nonparametric test) and found that there is a difference between the distributions of the running times with a significance value of 0.003.

The above strongly indicates that a big data system's performance is affected by:

- architecture – the number of files intended to be ingested and their volume
- configuration – block size limitation setting.

The understanding of the relationship between the above and its effect on performance is crucial, and if not designed correctly, loss of resources will most likely occur, including performance degradation.

As a preliminary outcome from this presentation, in some cases an organisation that uses big data platforms may examine the current block sizes configuration, based on a variety of parameters, such as: data capturing and storing demands, the organisation's infrastructure, type/volume/rate of the calculative data, etc.

It is important to mention that in order to achieve a greater understanding as to what extent the performance is affected, further research is needed, since:

1 big data environments are normally intended to ingest greater volumes of data

2 big data environments are normally intended to handle a greater number of sources, i.e., the number of files

3 real world applications are more complex than the code developed for the purpose of this research.

To this end, we intend to perform a more comprehensive research, taking into account the factors mentioned above. Such research will increase the understanding of the effects of a system's architecture and configuration on performance, while establishing a baseline supporting a big data environments' optimisation.

Acknowledgements

We would like to thank Mr. Ein-Dor Oded for his time and effort in reviewing and editorial commenting upon this paper to help improve the final product. Thank you.

References

Ackoff, R.L. (1989) 'From data to wisdom', *Journal of Applied Systems Analysis*, Vol. 16, No. 1, pp.3–9.

Ahad, M.A. and Biswas, R. (2018) 'Dynamic merging based small file storage (DM-SFS) architecture for efficiently storing small size files in Hadoop', *Procedia Computer Science*, Vol. 132, pp.1626–1635.

Andreolini, M., Colajanni, M., Pietri, M. and Tosi, S. (2015) 'Adaptive, scalable and reliable monitoring of big data on clouds', *Journal of Parallel and Distributed Computing*, Vols. 79–80, pp.67–79.

Assunção, M.D., Calheiros, R.N., Bianchi, S., Netto, M.A. and Buyya, R. (2015) 'Big data computing and clouds: trends and future directions', *Journal of Parallel and Distributed Computing*, Vols. 79–80, pp.3–15.

Bansal, S.K. (2014) 'Towards a semantic extract-transform-load (ETL) framework for big data integration', *IEEE International Congress on Big Data (BigData Congress)*, June, pp.522–529), June.

Beyer, M. (2011) *Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data*, Gartner, Archived from the original, Vol. 10.

Chandrasekar, S., Dakshinamurthy, R., Seshakumar, P. G., Prabavathy, B. and Babu, C. (2013) 'A novel indexing scheme for efficient handling of small files in hadoop distributed file system', *International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, January, pp.1–8.

Dean, J. and Ghemawat, S. (2008) 'MapReduce: simplified data processing on large clusters', *Communications of the ACM*, Vol. 51, No. 1, pp.107–113.

Dhawan, S. and Rathee, S. (2013) 'Big data analytics using Hadoop components like pig and hive', *American International Journal of Research in Science, Technology, Engineering & Mathematics*, Vol. 88, No. 1, pp.13–131.

Engelberg, G., Koren, O. and Perel, N. (2016) 'Big data performance evaluation analysis using apache pig', *International Journal of Software Engineering and Its Applications*, Vol. 10, No. 11, pp.429–440.

Ghemawat, S., Gobioff, H. and Leung, S.T. (2003) 'The Google file system', *SOSP'03: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, ACM Press, New York, NY, USA, pp.29–43.

IBM (2017) *10 Key Marketing Trends for 2017 and Ideas for Exceeding Customer Expectations* [online] http://comsense.consulting/wp-content/uploads/2017/03/10_Key_Marketing_Trends_for_2017_and_Ideas_for_Exceeding_Customer_Expectations.pdf (accessed August 2019).

Jach, T., Magiera, E. and Froelich, W. (2015) 'Application of HADOOP to store and process big data gathered from an urban water distribution system', *Procedia Engineering*, Vol. 119, pp.1375–1380.

Katal, A., Wazid, M. and Goudar, R.H. (2013) 'Big data: issues, challenges, tools and good practices', *Sixth International Conference on n Contemporary Computing (IC3)*, IEEE, pp.404–409.

Kendal, D., Koren, O. and Perel, N. (2016) 'Pig vs. hive use case analysis', *International Journal of Software Engineering and Its Applications*, Vol. 9, No. 12, pp.267–276.

Khan, N., Yaqoob, I., Hashem, I.A.T., Inayat, Z., Mahmoud Ali, W.K., Alam, M. and Gani, A. (2014) 'Big data: survey, technologies, opportunities, and challenges', *The Scientific World Journal*, p.8.

Li, G., Ooi, B.C., Feng, J., Wang, J. and Zhou, L. (2008) 'EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data', *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ACM, pp.903–914.

Loebman, S., Nunley, D., Kwon, Y., Howe, B., Balazinska, M. and Gardner, J.P. (2009) 'Analyzing massive astrophysical datasets: can pig/Hadoop or a relational DBMS help?', *IEEE International Conference on Cluster Computing and Workshops, 2009. CLUSTER'09*, IEEE, pp.1–10.

Nghiem, P.P. and Figueira, S.M. (2016) 'Towards efficient resource provisioning in MapReduce', *Journal of Parallel and Distributed Computing*, Vol. 95, pp.29–41.

Papadimitriou, S. and Sun, J. (2008) 'Disco: distributed co-clustering with map-reduce: a case study towards petabyte-scale end-to-end mining', *Eighth IEEE International Conference on Data Mining, 2008. ICDM'08*, IEEE, December pp.512–521.

Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., DeWitt, D.J., Madden, S. and Stonebraker, M. (2009) 'A comparison of approaches to large-scale data analysis', *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, pp.165–178, ACM.

Reuther, A., Byun, C., Arcand, W., Bestor, D., Bergeron, B., Hubbell, M., Jones, M., Michaleas, P., Prout, A., Rosa, A. and Kepner, J. (2018) 'Scalable system scheduling for HPC and big data', *Journal of Parallel and Distributed Computing*, Vol. 111, pp.76–92.

Siddiqa, A., Karim, A. and Gani, A. (2017) 'Big data storage technologies: a survey', *Frontiers of Information Technology & Electronic Engineering*, Vol. 18, No. 8, pp.1040–1070.

Stewart, R.J., Trinder, P.W. and Loidl, H.W. (2011) 'Comparing high level mapreduce query languages', *International Workshop on Advanced Parallel Processing Technologies*, pp.58–72, Springer Berlin, Heidelberg.

Storey, V.C. and Song, I.Y. (2017) 'Big data technologies and management: what conceptual modeling can do', *Data & Knowledge Engineering*, Vol. 108, pp.50–67.

Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P. and Murthy, R. (2009) 'Hive: a warehousing solution over a map-reduce framework', *Proceedings of the VLDB Endowment*, Vol. 2, No. 2, pp.1626–1629.

Wang, J., Wu, H. and Wang, R. (2017) 'A new reliability model in replication-based big data storage systems', *Journal of Parallel and Distributed Computing*, Vol. 108, pp.14–27.

Warden, P. (2011) *Big Data Glossary*, O'Reilly Media Inc., Cambridge, MA 02138, USA.

White, T. (2015) *Hadoop: The Definitive Guide*, 4th ed., O'Reilly Media Inc., Sebastopol, CA, USA.

Wu, W., Lin, W., Hsu, C.H. and He, L. (2018) 'Energy-efficient Hadoop for big data analytics and computing: a systematic review and research insights', *Future Generation Computer Systems*, Vol. 86, pp.1351–1367.

Notes

1 https://www.apache.org/.

2 http://hadoop.apache.org/.

3 http://nutch.apache.org/.

4 https://www.yahoo.com/.

5 http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html.

6 http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

7 http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

8 http://hive.apache.org/.

9 https://www.facebook.com/.

10 http://pig.apache.org/.

11 http://hive.apache.org/.

Appendix

A1 Tests plan

Table 9 Tests plan

Test #	*Scenario #*	*# of files*	*Dataset volume (MB)*	*# of blocks used*	*Test #*	*Scenario #*	*# of files*	*Dataset volume (MB)*	*# of blocks used*
1	A	1	360	3	31	B	2	180	4
2	A	1	360	3	32	B	2	180	4
3	A	1	360	3	33	B	2	180	4
4	A	1	360	3	34	B	2	180	4
5	A	1	360	3	35	B	2	180	4
6	A	1	360	3	36	B	2	180	4
7	A	1	360	3	37	B	2	180	4
8	A	1	360	3	38	B	2	180	4
9	A	1	360	3	39	B	2	180	4
10	A	1	360	3	40	B	2	180	4
11	A	1	360	3	41	C	3	120	3
12	A	1	360	3	42	C	3	120	3
13	A	1	360	3	43	C	3	120	3
14	A	1	360	3	44	C	3	120	3
15	A	1	360	3	45	C	3	120	3
16	A	1	360	3	46	C	3	120	3
17	A	1	360	3	47	C	3	120	3
18	A	1	360	3	48	C	3	120	3
19	A	1	360	3	49	C	3	120	3
20	A	1	360	3	50	C	3	120	3
21	B	2	180	4	51	C	3	120	3
22	B	2	180	4	52	C	3	120	3
23	B	2	180	4	53	C	3	120	3
24	B	2	180	4	54	C	3	120	3
25	B	2	180	4	55	C	3	120	3
26	B	2	180	4	56	C	3	120	3
27	B	2	180	4	57	C	3	120	3
28	B	2	180	4	58	C	3	120	3
29	B	2	180	4	59	C	3	120	3
30	B	2	180	4	60	C	3	120	3

A2 Test results

Table 10 Test results

Scenario #	*Test #*	*Job*	*Job run time (mm:ss.000)*	*Scenario #*	*Test #*	*Job*	*Job run time (mm:ss.000)*
A	1	1	00:23.0	B	31	1	00:38.8
		2	00:29.9			2	00:16.2
	2	1	00:34.5		32	1	00:22.7
		2	00:28.3			2	00:27.5
	3	1	00:23.0		33	1	00:32.4
		2	00:28.1			2	00:20.4
	4	1	00:31.7		34	1	00:23.7
		2	00:31.3			2	00:23.6
	5	1	00:23.5		35	1	00:24.7
		2	00:28.2			2	00:29.0
	6	1	00:23.1		36	1	00:29.9
		2	00:21.2			2	00:29.1
	7	1	00:22.7		37	1	00:40.0
		2	00:34.0			2	00:23.8
	8	1	00:23.7		38	1	00:46.2
		2	00:21.2			2	00:40.0
	9	1	00:24.5		39	1	00:32.7
		2	00:24.1			2	00:40.4
	10	1	00:23.6		40	1	00:39.8
		2	00:19.2			2	00:32.8
	11	1	00:22.5	C	41	1	00:24.4
		2	00:17.3			2	00:30.3
	12	1	00:31.6		42	1	00:18.3
		2	00:21.1			2	00:19.7
	13	1	00:38.8		43	1	00:23.3
		2	00:41.5			2	00:21.2
	14	1	00:21.3		44	1	00:29.9
		2	00:26.6			2	00:40.9
	15	1	00:22.5		45	1	00:22.1
		2	00:24.9			2	00:22.0
	16	1	00:28.6		46	1	00:35.0
		2	00:25.8			2	00:44.0
	17	1	00:23.4		47	1	00:32.2
		2	00:50.5			2	00:37.8
	18	1	00:33.2		48	1	00:34.8
		2	01:01.2			2	00:21.6

Table 10 Test results (continued)

Scenario #	*Test #*	*Job*	*Job run time (mm:ss.000)*	*Scenario #*	*Test #*	*Job*	*Job run time (mm:ss.000)*
A	19	1	00:22.4	C	49	1	00:29.7
		2	00:23.2			2	00:48.3
	20	1	00:24.7		50	1	00:29.2
		2	00:47.4			2	00:19.5
B	21	1	00:26.6		51	1	00:23.1
		2	00:24.6			2	00:26.0
	22	1	00:32.1		52	1	00:24.6
		2	00:28.9			2	00:23.9
	23	1	00:24.1		53	1	00:37.7
		2	00:23.3			2	00:37.3
	24	1	00:27.8		54	1	00:34.4
		2	00:19.9			2	00:19.8
	25	1	00:23.2		55	1	00:30.0
		2	00:28.2			2	00:31.1
	26	1	00:32.2		56	1	00:23.8
		2	00:43.1			2	00:31.5
	27	1	01:00.9		57	1	00:29.7
		2	00:39.2			2	00:20.1
	28	1	00:39.5		58	1	00:23.5
		2	00:35.2			2	00:42.2
	29	1	00:48.1		59	1	00:34.4
		2	00:50.8			2	00:53.4
	30	1	00:51.1		60	1	00:33.0
		2	00:32.7			2	00:20.1