MRbox: Simplifying Working with Remote Heterogeneous Analytics and Storage Services via Localised Views

Athina Kyriakou
National Technical University of Athens
Zografou, Greece
athina.skyriakou@gmail.com

Iraklis A. Klampanos
National Centre for Scientific Research "Demokritos"
Agia Paraskevi, Greece
iaklampanos@iit.demokritos.gr

ABSTRACT

The management, analysis and sharing of big data usually involves interacting with multiple heterogeneous remote and local resources. Performing data-intensive operations in this environment is typically a non-automated and arduous task that often requires deep knowledge of the underlying technical details by non-experts. MapReduce box (MRbox) is an open-source experimental application that aims to lower the barrier of technical expertise needed to use powerful big data analytics tools and platforms. MRbox extends the Dropbox interaction paradigm, providing a unifying view of the data shared across multiple heterogeneous infrastructures, as if they were local. It also enables users to schedule and execute analytics on remote computational resources by just interacting with local files and folders. MRbox currently supports Hadoop and ownCloud/B2DROP services and MapReduce jobs can be scheduled and executed. We hope to further expand MRbox so that it unifies more types of resources, and to explore ways for users to interact with complex infrastructures more simply and intuitively.

1 INTRODUCTION

A number of multidisciplinary scientific and business problems require the use of advanced analytics tools and the management, processing and sharing of big data across local and remote platforms, services and cloud infrastructures. Users often define complex analytics pipelines using workflow management systems, which can also orchestrate their execution. However, in practice, this orchestration is only partially automatic because the tools, resources and execution environments used by different organisations are highly heterogeneous while the majority of workflow management systems provide solutions to field-specific problems [8]. As a result, researchers and data analysts are forced to encounter the technical details of the underlying tools and resources.

In response to this challenge, MRbox attempts to lower the barrier of required technical knowledge to use the needed tools and infrastructures and to hide the complexity of big data management from non-experts. To this end, we investigate providing local views on remote computation and file storage cloud resources, extending the paradigm of cloud storage, synchronisation and data exchange services, such as Dropbox [5]. Using the file system of the local operating system, MRbox provides an overview of the data stored in the connected remote infrastructures and enables users to perform computation jobs on them, as if they were local, and to share the files produced easily.

More specifically, users can delete, modify or move data sets, just by interacting with local files and folders and without the need to know explicitly in which remote or local resource they reside. In addition, in the current prototype, users have the ability to schedule and execute MapReduce jobs [3] on a remote Hadoop cluster [12] from their local machine. The input data can be stored in any connected resource but users do not have to run specialised HDFS commands [12] on their terminal to move them to a specific Hadoop cluster and to issue the MapReduce job. Lastly, output data generated from MapReduce processes are fetched locally for ad hoc analytics and they are pushed onto the B2DROP data exchange service[6], so that users can share their results with colleagues and with the outside world.

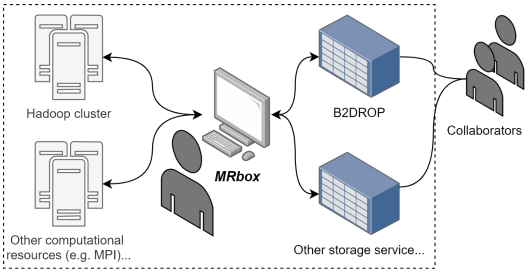

Figure 1: The concept of MRbox is to simplify interacting with multiple analytics and big data resources by extending the cloud storage paradigm.

MRbox could be useful to different actors. It can be used by (i) data analysts, researchers and engineers working on big data crunching problems, but also by (ii) systems and e-infrastructures that seek to seamlessly integrate with third-party big data and data management resources. MRbox is an open-source project hosted on GitHub[1].

2 ASSUMPTIONS AND USE CASES

For the development of MRbox, the two main assumptions made are: (i) Users do not necessarily have full control of the needed remote resources. They just have rights to create, delete, modify and relocate files in their remote folders and run computation jobs on them. (ii) When working with large and complex data sets, moving them should be avoided unless necessary[2].

For the development of the current prototype, the following use cases were identified. Firstly, users should easily connect to the integrated remote infrastructures, just by running the MRbox application. Secondly, they need to have a complete view of the file system hierarchy in the remote resources from their local file system. When a remote data set exceeds in size the maximum file size that can be replicated in the local machine, a `link` file will be created locally, containing only the path to the remote file. Thirdly, users of MRbox would have a live preview of the files residing in a remote source by running file system commands in their terminal (e.g. `head`, `tail`), even when the files do not exist

[1] https://github.com/AthinaKyriakou/mrbox

physically in the local machine. Moreover, users should be able to schedule computation jobs that will be executed in remote processing infrastructures by referring to files as if they were local. Lastly, users need to have seamless access to the output files generated by the scheduled jobs in order to do further analysis, or share them with their peers.

3 SYSTEM OVERVIEW

This section summarises the main components and features of the application.

3.1 A Sample Session With and Without MRbox

As an example, let us consider the scheduling of a MapReduce job, which is the currently supported computation framework. In the table below we compare the set of actions that a user needs to perform, with and without using MRbox.

Using MRbox	Without Using MRbox
1. Code the Map and Reduce functions locally.	1. Code the Map and Reduce functions locally.
2. Move the input file (or a `link` to it) in the local `MRbox` folder, if it is not already there.	2. If the input data set is in the local file system, copy it to an HDFS cluster via HDFS terminal commands. If it is on a remote resource, use the resource-specific API or instructions to copy it to HDFS.
3. Create a `yaml` file specifying the local paths to Map and Reduce functions, input file and desired output location. Move the `yaml` file in the local `MRbox` folder.	3. Use the Hadoop Streaming API to run the job. Move and generally manage the files generated using the HDFS commands as needed.
Outputs are automatically synced in the local `MRbox` folder and in B2DROP. Links are created if the files are larger than a predefined size.	Use HDFS terminal commands to fetch the outputs locally. Use an ownCloud client or API to copy them to B2DROP.

3.2 Supported Resources

MRbox currently assumes a UNIX clone as the local and host system and supports the Apache Hadoop framework [13] and the B2DROP service [6].

Apache Hadoop is a widely used and open-source MapReduce framework. It includes two main modules; Hadoop Distributed File System (HDFS) and the Hadoop MapReduce. HDFS is the distributed file system primarily used by Hadoop applications, providing high data-access performance, fault tolerance and native support of large data sets. MRbox establishes a connection to the desired HDFS on a remote cluster and it interacts with Hadoop MapReduce via the Hadoop command line tools provided.

B2DROP[6] is a user-friendly data synchronisation and exchange service created by the European Data Infrastructure (EUDAT[2]) for research communities. It provides a secure storage environment for long-tail but still volatile data that are subject to active research, while facilitating the process of sharing and keeping them up-to-date across different machines. It offers up to 20GB of storage per user. B2DROP offers automatic synchronisation via ownCloud[3], an open source file synchronisation and sharing tool. Alternatively, the service can be accessed on the Web via an intuitive user interface, or on the local machine via a WebDAV Client.

3.3 MRbox Configuration

To connect to a remote HDFS, the local host needs to be configured as a client node. This usually requires to have a copy of the Hadoop distribution locally available and configured to access the remote cluster[15]. In `MRbox.conf` – the configuration file of MRbox – the user needs to specify the host and port of the HDFS NameNode, as well as the path to the local Hadoop installation. The user can also determine the absolute path of the local and HDFS folders that will be in sync, and the maximum size of the files that will be retrieved locally once created in an integrated remote resource. Depending on the distribution policy for MRbox, the Hadoop part of the configuration may also be pre-configured, therefore saving the end user from even being aware of the connection specifics to the remote Hadoop cluster.

In addition, the B2DROP service is currently used by installing on the local machine the ownCloud Desktop Synchronisation Client following the B2DROP documentation. Both B2DROP and MRbox monitor the same local folder. A more fine-grained integration of ownCloud in MRbox will be developed in the future, e.g. by modifying the existing client.

3.4 Managing Files

MRbox creates folders locally and on the integrated remote resources, under the root paths specified in the `MRbox.conf` file. In order to keep track of local and remote instances of files and to promote consistency, MRbox implements a catalogue. This section describes this catalogue, how it stores mappings of local paths to remote ones, the synchronisation process between the local `MRbox` folder, HDFS and B2DROP, the implementation of links to comply with local size constraints, and the use of checksums for data validation.

3.4.1 The Local Catalogue. The local catalogue is used to maintain mappings between local and remote files and directories. In the current prototype of MRbox, the local catalogue maps the local paths to the paths of the corresponding files and directories on HDFS, making it possible for the user to interact with the HDFS of a remote cluster through the local file system hierarchy. Its current implementation is in SQLite[10], a lightweight, self-contained SQL database engine that does not require a separate server process to operate and therefore is integrated in the application itself. For each file and folder, a record is created in the database consisting of the path, the most recent modification time and the checksum of the local and respective copy on HDFS. Each record also has an attribute specifying whether the local object is a file, a link (see Section 3.4.3) or a directory.

3.4.2 Synchronisation. Synchronisation between the local `MRbox` folder and HDFS is one-way as we can only monitor the local folder for changes. Our current implementation makes use of the Python Watchdog library [11], which monitors a designated local folder for changes. Depending on the type of the event identified, one of the following actions will take place:

- `on_created()`: When a file, link or directory is created locally, it is registered in the local catalogue. For the case

[2] https://eudat.eu/

[3] https://owncloud.com/

of file or directory, a copy is created on the remote HDFS. If the file created has a `yaml` extension, MRbox attempts to run a MapReduce job on the remote Hadoop cluster, according to the specification passed in the `yaml` file (Figure 2).
- `on_deleted()`: When a file, link or directory is deleted locally, MRbox deletes the corresponding HDFS file and the record from the local catalogue.
- `on_modified()`: When a file is modified locally, the HDFS file is modified accordingly and the local and HDFS checksums on the local catalogue are updated.
- `on_moved()`: When a file, link or directory is moved within the local `MRbox` folder, the corresponding file or directory is relocated accordingly within the HDFS folder.

Handling Bidirectional Synchronisation. As discussed above, the current prototype assumes that users do not necessarily have full control over the remote resources. This means that, if files on HDFS change by any means other than through interacting with MRbox, MRbox will not have a way to track these changes, therefore leading to inconsistencies. To handle such potential inconsistencies between the local and HDFS `MRbox` folders an offline synchronisation process should be scheduled to run periodically. This is left for future work as low priority, since MRbox's main goal is to provide local views on remote HDFS resources, hiding direct access from its users.

In the case of B2DROP, bidirectional synchronisation is achieved through its desktop client, which is installed locally (Figure 2). To support special functionalities of MRbox (e.g. see Section 3.4.3), this client would have to be modified further.

3.4.3 Links. In MRbox, links are a special read-only file type recognised by the `link` extension and registered in the local catalogue. Links are created to comply with local file size restrictions when large data sets are generated on the HDFS cluster and need to be made available locally. In the current prototype such files are the outputs of MapReduce jobs on HDFS.

The local file size limit can be specified in the MRbox configuration. After the completion of a MapReduce job, the local file size limit is compared against the size of the output file produced on HDFS. If the generated file is larger than the maximum allowable size limit, a `link` file is created, which contains the path to the remote file. This link can also be used in the browser to view the file on HDFS, if this is supported by the remote cluster. Moreover, if a `link` is deleted or moved within the local `MRbox` folder, the corresponding remote file will be deleted or moved respectively.

Synchronisation of Links Between B2DROP and HDFS. In the current prototype, a link file in the local MRbox folder is synchronised verbatim onto B2DROP. However, this is not always desirable, since users would expect the complete file to be available on a data exchange resource such as B2DROP. After all, the users that need to access a file on B2DROP may not have access to the HDFS cluster where it was created. To allow for special synchronisation treatment for the case of large files that appear as links locally, a customised synchronisation client would need to be implemented.

3.4.4 File Checksums. Checksums are used to guarantee data integrity in file transfers between the local and HDFS folder. HDFS uses a 32-bit cyclic redundancy check based on the Castagnoli polynomial (CRC32C). To perform end-to-end client side validation, MRbox adopts a composite CRC file checksum introduced in Apache Hadoop 3.1.1 that can be configured by setting `dfs.checksum.combine.mode` to `COMPOSITE_CRC`. The composite CRC checksum is not applicable to links and directories. In contrast to Hadoop's default MD5 of MD5 of CRC file checksum computed across chunks and blocks, the composite CRC is independent of chunk and block configurations and describes only the logical file contents. As a result, it permits comparison between striped and replicated files, between HDFS instances with potentially different chunk and block size configurations, as well as between HDFS and local files or other external storage systems that implement Hadoop's File System interface [7, 14]. At the end of each transfer, local file checksums are computed and compared against the checksums computed on HDFS. If the checksums do not match, the transfer is repeated.

3.5 Scheduling MapReduce Jobs

MRbox aims to highlight the functional aspects of MapReduce by hiding the technical implementation details from users. In the future, MRbox could be extended to support more computational frameworks. To trigger a MapReduce job users need to:

- Implement the Map and Reduce functions in two distinct files locally. Any programming language supported by the Hadoop Streaming utility can be used [16].
- Specify in a `yaml` file the absolute paths of the mapper and reducer scripts in the local file system. Also, users need to define the relative paths of the input as well as the output location. The input path can point to a file or a link. All paths are local and users do not need to know the file structure of the remote HDFS cluster.
- Move or save the `yaml` file in the local `MRbox` folder to trigger the execution.

MRbox will automatically issue the configured MapReduce job to the remote Hadoop using the Hadoop Streaming utility. The outputs will be fetched locally in the specified output path as files or links. The ownCloud Desktop Client, will replicate the output file to B2DROP, giving the user the possibility to share it with people working on the same project.

3.6 Additional Tools

The current prototype enables users to get a live preview of data sets residing on HDFS, by running `mrview.py cmd path` for a file, link or directory in the local `MRbox` folder. If the specified path corresponds to a file or directory, the supported UNIX command is executed. If the path corresponds to a link, the data set does not exist locally due to size constraints. Users can run `head` and `tail` commands to get an overview of the HDFS data corresponding to the link. The list of supported commands for links can be extended to a complete suite of tools in the future.

4 RELATED WORK

Scientific and business workflow management systems and Workflow-as-a-Service platforms [1, 4, 9, etc.] often facilitate the processing of big data across multiple e-infrastructures. However, only a limited number of them integrate big data frameworks, such as Hadoop, directly, without focusing on a certain data organisation or field-specific problems [17]. In addition, these systems still present challenges for big data analytics in the cloud and when used across organisations with potentially heterogeneous resources and execution environments [8]. As a result, the data operations that need to be performed are partially automated and professionals have to manually configure and use the involved services and local and remote infrastructures. Finally,

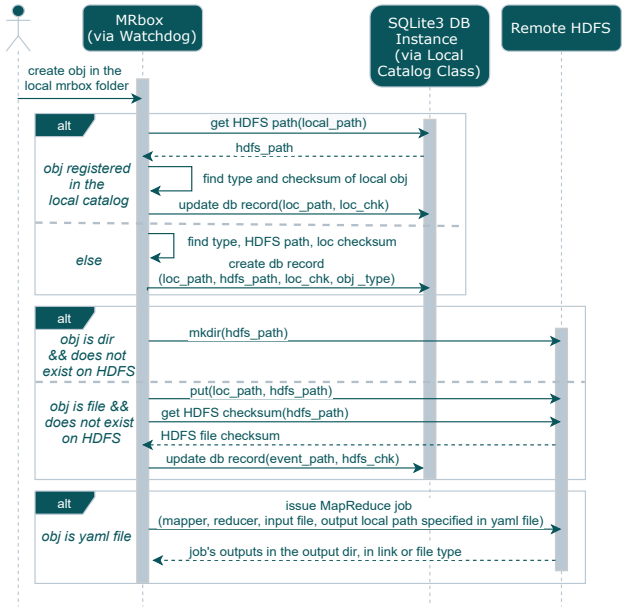

Figure 2: Sequence of events after a file is created in the local `MRbox` folder

identifying whether a workflow supports a specific data lake, platform or tool is usually a challenging task and users need to get familiar with a new Graphical User Interface (GUI) and the specific workflow model to efficiently use the management system [8, 17]. In contrast to workflow management systems, MRbox does not require that its users learn a new GUI or a workflow model specification language.

Furthermore, MRbox borrows from and extends cloud storage, synchronisation and data exchange services, such as Dropbox. However, apart from synchronising files and folders and allowing users to share their data, MRbox investigates the extension of this interaction paradigm for data analytics and distributed processing, as well as for synchronising amongst multiple resources.

5 CONCLUSIONS AND FUTURE WORK

For the processing and sharing of big data, researchers and data analysts use a plethora of tools and heterogeneous infrastructures. As a result, performing data-intensive tasks is typically non-automated and arduous and requires technical knowledge of the underlying systems. MRbox aims to lower the barrier of technical expertise by hiding the complexity of big data technologies from non-experts. Using the file system hierarchy of the local operating system, MRbox provides a unifying view of local and remote resources and the data residing in them, and enables users to schedule computational jobs on remote infrastructures as if they were local. In the current prototype, Hadoop and B2DROP services are supported and MapReduce jobs can be scheduled.

This work can be improved in several ways. Firstly, we need to test MRbox under heavier workloads. A more fine-grained integration of ownCloud in MRbox will be implemented to enable users to directly connect to the B2DROP service without installing a separate client. Secondly, the ownCloud client needs to be modified and expanded further to support special functionalities of MRbox, e.g. for the case of links, the complete data set should be available on resources such as B2DROP. Thirdly, according to a distribution policy, the connection to remote infrastructures, such as a Hadoop cluster, could be pre-configured, saving the end user from even being aware of the configuration specifics. Additionally, to verify our concept, we intend to measure the usability and performance of the current prototype via user studies covering a range of uses and workloads. Furthermore, while maintaining simplicity, we will investigate expanding to other resources and execution contexts, e.g. triggering the execution of numerical codes on MPI clusters.

A valuable more theoretical follow-up work could be the formalisation of resource types and integration policies, which will add to the extensibility of MRbox while minimising the risk of data inconsistencies and resource mismanagement. This could lead to future research towards a more general framework, to describe arbitrary computational and storage resources. An alternative direction could be the exploitation of the unifying file-based view of heterogeneous resources to integrate widely used tools and user interfaces, e.g. spreadsheet applications to transparently make use of large data sets on remote HDFS clusters.

REFERENCES

[1] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. 2004. Kepler: an extensible system for design and execution of scientific workflows. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.* 423–424. https://doi.org/10.1109/SSDM.2004.1311241

[2] Malcolm Atkinson. 2018. Pushing the Limits of Data Powered Research. https://doi.org/10.5281/zenodo.1164420

[3] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (2008), 107–113.

[4] Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, and Kent Wenger. 2015. Pegasus: a Workflow Management System for Science Automation. *Future Generation Computer Systems* 46 (2015), 17–35. https://doi.org/10.1016/j.future.2014.10.008 Funding Acknowledgements: NSF ACI SDCI 0722019, NSF ACI SI2-SSI 1148515 and NSF OCI-1053575.

[5] Dropbox [n.d.]. *What is Dropbox - Features Overview - Dropbox.* Dropbox. https://www.dropbox.com/features.

[6] EUDAT Collaborative Data Infrastructure 2019. *B2DROP User Documentation.* EUDAT Collaborative Data Infrastructure. https://eudat.eu/services/userdoc/b2drop.

[7] Google Cloud 2020. *Validating data transfers between HDFS and Cloud Storage.* Google Cloud. https://cloud.google.com/solutions/migration/hadoop/validating-data-transfers.

[8] Samiya Khan, Syed Arshad Ali, Nabeela Hasan, Kashish Ara Shakil, and Mansaf Alam. 2019. Big data scientific workflows in the cloud: Challenges and future prospects. In *Cloud computing for geospatial big data analytics.* Springer, 1–28.

[9] Iraklis A. Klampanos, Chrysoula Themeli, Alessandro Spinuso, Rosa Filgueira, Malcolm Atkinson, André Gemünd, and Vangelis Karkaletsis. 2020. DARE Platform a Developer-Friendly and Self-Optimising Workflows-as-a-Service Framework for e-Science on the Cloud. *Journal of Open Source Software* 5, 54 (2020), 2664. https://doi.org/10.21105/joss.02664

[10] Python Software Foundation 2020. *sqlite3 — DB-API 2.0 interface for SQLite databases.* Python Software Foundation. https://docs.python.org/3/library/sqlite3.html.

[11] Python Software Foundation 2020. *watchdog 1.0.2.* Python Software Foundation. https://pypi.org/project/watchdog/.

[12] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. 2010. The Hadoop Distributed File System. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST).* IEEE, 1–10.

[13] The Apache Software Foundation [n.d.]. *Apache Hadoop.* The Apache Software Foundation. https://hadoop.apache.org/.

[14] The Apache Software Foundation 2019. *Expose file-level composite CRCs in HDFS which are comparable across different instances/layouts.* The Apache Software Foundation. https://issues.apache.org/jira/browse/HDFS-13056.

[15] The Apache Software Foundation 2019. *Hadoop: Setting up a Single Node Cluster.* The Apache Software Foundation. https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html.

[16] The Apache Software Foundation 2019. *Hadoop Streaming.* The Apache Software Foundation. https://hadoop.apache.org/docs/r1.2.1/streaming.html.

[17] Jianwu Wang, Daniel Crawl, and Ilkay Altintas. 2009. Kepler + Hadoop: A General Architecture Facilitating Data-Intensive Applications in Scientific Workflow Systems. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science.* 1–8.