Distributed Resource Management in Operating Systems: A Case Study on HDFS and YARN

Sivaprasath R.[1], Vedeshvar L.[2], Bharath M. D.[3], Achari Magesh[4], Anisha C. D.[5]

[1-4]Student, [5]Assistant Professor, Department of Computer Science and Engineering, PSGCT, India.

E-mail: [5]ani.c.dass@gmail.com

Abstract

This research study focuses on analysing the role of distributed resource management in enhancing the scalability and reliability of the linked systems. This study presents a detailed analysis on the architectures, benefits, and inherent drawbacks of the Hadoop Distributed File System (HDFS) and Yet Another Resource Negotiator (YARN). YARN offers flexible resource scheduling through Fair and Capacity schedulers, while HDFS offers fault-tolerant, scalable storage through a block-based, replicated, and locality-optimized design. Although robust, limitations like resource contention in YARN and the Name Node's single point of failure in HDFS still exist. In order to address the evolving challenges in modern computing, this study also explores the potential research domains like serverless architecture for dynamic scaling, latency-conscious edge computing, and AI-based resource forecasting.

Keywords: Distributed resource management, HDFS, YARN, Edge computing.

1. Introduction

Distributed systems are a key element in today's computing, which we see to scale and dependably in real time, cloud services, and big data analysis. They enable smooth operation in many dynamic and diverse environments by establishing interconnection among many nodes. Also, in the area of distributed resource management, there is an increasing need to manage the computational and storage resources to maintain performance, scalability and reliability, which plays a major role in distributed systems.

In the field of Resource Management, we see the advantages of using both Yet Another Resource Negotiator (YARN) and Hadoop Distributed File System (HDFS). HDFS is a base component in the field of large scale distribution that uses a master slave structure to present to users a storage solution for large sets of data across many nodes. Features like block level storage, data replication and locality optimization to reduce network overhead and increase performance by enhancing the fault tolerance and dependability of HDFS. Also in HDFS' advantage are its features but it does present some issues which include a single point of failure in the Name Node, which has seen the development of high availability solutions. YARN is the resource management and scheduling framework of Hadoop which puts forward an effective way to use cluster resources. By separating resource management from application logic, YARN gives us what we need in terms of flexibility and scalability by using the architecture of Central Resource Manager and Per node NodeManagers. While YARN performs well in the area of resource management it also has many issues in terms of multi-tenant environments, which include resource management and communication overhead. This study looks at what is put forth in terms of technology and trends that have the chance to modify the HDFS and YARN models. The recent introduction of AI driven systems can enable real time resource allocation and anomaly detection. We also have serverless architectures which are all about dynamic scaling with idle resource use and at the same time Edge Computing enables new 1.6.5 paradigms for light weight and low latency resource management. This research study presents the value of the aforementioned models in terms of flexible, reliable and sustainable solutions in order to guarantee the performance and dependability of distributed resource management.

1.1 Major Contributions of the Research

This research study contributes to the field of distributed resource management in many ways. We look at related issues of resource contention and system heterogeneity along with basic concepts of scalability, fault tolerance, efficient resource allocation and data locality. We look at the Hadoop ecosystem in depth which we study how the HDFS and YARN architectures perform in terms of distributed storage and scheduling of workloads. Also we include in the discussion the recent innovations in serverless architectures, edge computing and AI enabled resource allocation. We also present a structured review of different distributed resource management strategies in Table I, which includes the process, performance analysis, advantages and disadvantages of different models.

We present a comparison on various scheduling policies (for example Capacity Scheduler and Fair Scheduler) at different workloads and resource demands. To show out how scheduling policies play into resource allocation optimization we present an analysis on performance metrics like job completion time, resource utilization, and throughput. Also we present technical insight and put forth recommendations for what may improve future enhancements to the systems' adaptability and overall efficiency which in turn we also look at the hadoop distributed file system and the resource manager yarn do in different workloads.

This study reports on the results of our quantitative analysis which in addition to qualitative did also study HDFS, and YARN performance in detail using simulation. As to what we looked at for each policy's performance we looked at throughput, resource use, and job completion time. The empirical data from the study support out analysis and also add a practical dimension to how different scheduling policies and architectural elements play out in a variety of workloads.

2. Related Works

Resource management is a key element in the performance and scale up of distributed computing which in turn includes mobile edge computing (MEC), cloud infrastructures, and big data systems. Zhang and Debroy [1] report in detail on resource management in MEC which they look at in terms of latency issues, heterogeneity, and dynamic workloads. Also in this space, Huang, He, and Miao [2] study resource management in multi-tier web applications, which they look at through the lens of elastic mechanisms to adapt to the changing demands. From a different perspective, Moreira and Naik [3], [10] look at dynamic resource management within reconfigurable applications and lay out the base concepts for adaptive systems. Also, in high performance distributed computing, Hussain et al. [4] presented an in-depth study of resource allocation in terms of the trade-off between efficiency and computational overhead.

The Internet of Things (IoT) adds a layer of complexity. In terms of real time issues and resource constraints in pervasive IoT systems research is presented by Zahoor and Mir [5] which also brings out the need for lightweight and intelligent resource handling mechanisms. In large scale datasets' environment, Hadoop remains as the main focus. Research has been done on the Hadoop workloads' issue, which includes the work of job scheduling frameworks by Cheng et al. [6] and in depth usage info by White [7]. Also, Krauter, Buyya, and Maheswaran [8] present a classification of grid resource management which in turn is a base for current frameworks. At Facebook they did real time Hadoop processing which as reported by Borthakur et al. [9] shows how in practice real world applications differ from what is seen in theory due to production scale issues.

In the field of cloud computing Chen et al. [11] report on ALBERT which is a machine learning based resource manager for Hadoop workloads that they put forth which also presents how AI plays a role in the optimization of cloud resources. In terms of file systems, Huang et al. [12] report on I/O proportionality at the container level and also shown the increasing degree of control in resource management. What we see is that scheduling has been the focus of much research from the initial reports by Rao and Reddy [13] of studies done on MapReduce in cloud settings to the introduction by Yao et al. [14] of new algorithms for YARN clusters which in turn do not only report results but also put forth methods to improve performance and resource use. Also, Van Do et al. [15] pay close attention to issues of data rate in Hadoop based systems which they present as very complex operationally.

Table 1. Summary of Distributed Resource Management Approaches

Author/Year	**Strategies Used**	**Performance**	**Metrics**	**Merits**	**Demerits**
Zhang & Debroy (2023)	Survey on MEC resource management	Comprehensive but theoretical	Latency, scalability	Wide coverage of MEC topics	Lacks empirical validation
Huang et al. (2014)	Resource management in multi-tier apps	Effective in layered systems	Throughput, response time	Practical application insights	Outdated for current tech
Moreira & Naik (1997)	Reconfigurable application-based management	Adaptive in distributed systems	Utilization rate	Dynamic adaptability	Old hardware assumptions
Hussain et al. (2013)	Resource allocation survey for HPC	Broad scope	Efficiency, utilization	Extensive taxonomy	High-level, lacks case studies
Zahoor & Mir (2021)	IoT resource survey	Focus on IoT constraints	Energy, latency	Tailored to constrained devices	Lacks algorithmic detail
Cheng et al. (2015)	Deadline-aware scheduling in Hadoop	Improved job throughput	Makespan, deadline hit rate	Dynamic adaptability	Limited scalability data

White (2012)	Definitive Hadoop guide	N/A	N/A	Practical deployment reference	Not a research evaluation
Krauter et al. (2002)	Grid resource taxonomy	Categorical comparison	Adaptability, scalability	Foundational framework	Pre-cloud focus
Borthakur et al. (2011)	Real-time Hadoop at Facebook	Production-scale performance	Latency, data throughput	Real-world implementation	Specific to Facebook infrastructure
Chen et al. (2022)	Learning-based Hadoop resource management	Efficient optimization	Execution time, energy	AI integration	Training overhead
Huang et al. (2020)	IO sharing in big data systems	Balanced IO utilization	Proportionality, fairness	Improved data flow	Niche to specific FS types
Rao & Reddy (2012)	Improved MapReduce scheduling	Enhanced scheduling	Job time, resource usage	Focus on cloud Hadoop	Limited to MapReduce
Yao et al. (2019)	New Hadoop YARN schedulers	Better utilization	Resource efficiency	Cluster-wide optimization	Complexity in deployment
Van Do et al. (2015)	Data rate control in Hadoop	Better bandwidth control	Transfer rate, delay	Network-aware execution	No user-layer abstraction

3. Survey on Various Distributed Resource Management Approaches

A primary aspect of distributed resource management is dynamic resource allocation which in turn puts to use storage and processing power very well. In their 2022 work which also looked at compute offloading as a strategy for limited environs, Zhang and Debroy put forth the importance of real time resource allocation in mobile edge computing. Also they brought to fore that which which does the job of effective processing in resource starved systems is what which tasks are given dynamic priority. Also looked at is workload aware scheduling which in high performance distributed systems does what it does best by which it balances resource use across many workloads. What it does is it maximizes system performance by what it does to adapt to many computing needs. As for fault tolerance which is key to the issue of reliability in distributed systems, in 2023 Zahoor and Mir looked at fault tolerant mechanisms in IoT settings which they put forward to be lightweight redundancy and failover protocols. What these do is improve on reliability at the same time they take into the account the specific issues of IoT devices which include low compute power and energy efficiency. Also, in the area of scalability which is what we look to in growing systems, in 2014 Huang et al. put forth SLA based models for multi-tier web apps which did what it did best in balancing between performance and cost. What they did was to scale resources dynamically to meet service level agreements thus they in turn addressed the issues of complex layered architectures.

Distributed storage systems like Hadoop Distributed File System (HDFS) are key to fault tolerance and scale. Shvachko et al. (2010) put forth HDFS which includes block replication and data locality to improve storage performance and reliability in large scale settings. Borthakur (2013) reported on improvements we saw in high availability and snapshot features which in turn we solved for issues like single points of failure thus improving also what we see in terms of system reliability and recovery. YARN transformed resource management by putting the focus on separate resource allocation from application logic. Vavilapalli et al. (2013) reported on YARN's central Resource Manager and node Level Node Managers to enable better multi-tenant resource sharing. Also, in this time frame Zhao et al. (2016) did a research work on queueing policies, which included Fair and Capacity Schedulers to do better at what is fair and what is the use of the cluster. These worked to position YARN as a base element in distributed resource management frameworks. Also in today's computing, we have new paradigms like containerization and serverless computing, which are redefining distributed resource management. Abad et al. (2021) reviewed the role of Kubernetes in distributed systems to improve the modularity and fault isolation. Li et al. (2020) analyzed the issues of serverless computing, which include low latency scheduling and dynamic scaling. These studies point out the ongoing evolution of distributed resource management in response to the issues of current computing.

4. Case Study

At present we have 128 MB which are replicated across many Data Nodes for the purposes of fault tolerance and high availability. In the event of a failure, to bring back lost data the healthy nodes are used and perform re replication of blocks. Also, we see that data locality is a key element on which tasks are performed at the node which has the required data which in turn puts down network overhead and at the same time improves performance. Also, HDFS has a rack awareness feature which does a great job of putting data replica's out in different racks to improve fault isolation and data availability. This in turn means that should an entire rack go down our data is still available. But at the same time, we must note that we do have a single point of failure with the Name Node. In the case of a Name Node failure the whole HDFS cluster may go down and we lose access to our data which results in large scale unavailability. To that end we have seen the development of high availability options like the stand by Name Node and fail over systems which do a good job at smooth recovery from a Name Node failure. Though these do a great job they still have issues with sync and fail over which in turn play a role in the overall responsiveness of the system.

Overall HDFS is a reliable, fault tolerant and scalable which makes it a key element in big data processing. But we see that it has major issues in terms of single point of failure at the Name Node and the related failover problems. In the HDFS architecture as seen in Fig. 1 we have the Client, NameNode and DataNodes which are in communication. The NameNode's job is to manage metadata and the DataNodes which store and replicate the data blocks.

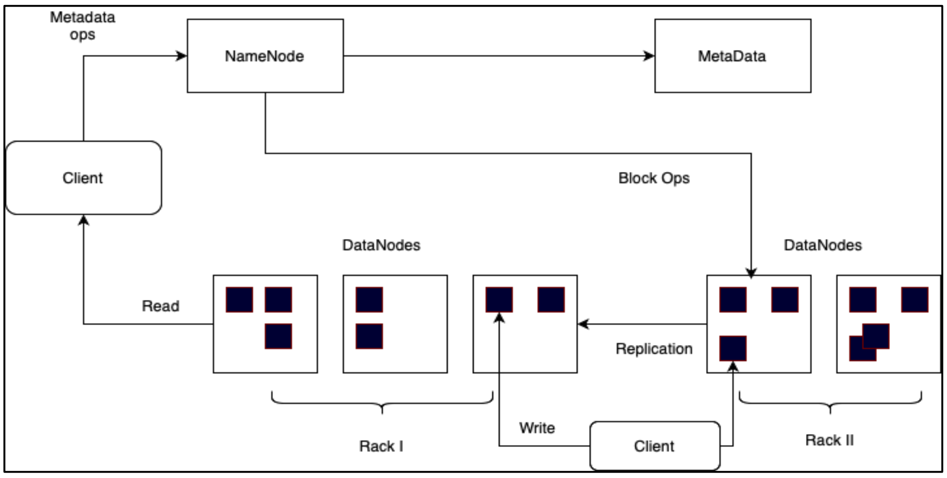

Figure 1. HDFS Architecture

Yet in the Hadoop environment YARN enables the heavy lifting of resource management and scheduling of diverse workloads across distributed clusters. YARN separates resource management from application logic which in turn gives us greater flexibility and scale. At its core is the Resource Manager, which is serves as the central authority for resource allocation. It uses advanced scheduling policies like Fair Scheduling which as the name suggests puts all users on an equal foot, and Capacity Scheduling which is a pre-determined set of priorities for different workloads. Also, it is able to do dynamic resource allocation, a key element in the design for multi-tenant success.

In each node, Node Managers track local resources and perform tasks within containers which are isolated environments that contain the needed resources (e.g., CPU, memory) for operation of particular applications. YARN has fault tolerant features like resubmitting tasks in case of container of node failures which in turn makes for what is minimal service interruption in the operation with which it does not go down. Also, it has a multi-tenancy ability, which is that it allows for multiple users and applications at the same time to make use of the cluster resources.

Even though this system is very flexible in design we do see issues with resource contention in high demand settings. When large scale resource requests are made at the same time the system may experience resource starvation which in turn causes delay for some jobs or in some cases, we may see that they just don't get scheduled at all. Also, in dynamic settings which include many different types of resources the issue is made more complex as our present allocation strategies may have trouble at the same time maintaining fair play and overall system efficiency.

Despite these issues, YARN's modular and scalable design which is what makes it the best choice for running into clusters of up to 1000 nodes at which also support both batch and real time processing. Also, from Fig 2 we see that the YARN architecture which shows how the Client Node puts in requests to the Resource Manager which in turn passes it to Node Managers in which they in turn see that the application is run in the containers across the networked nodes.

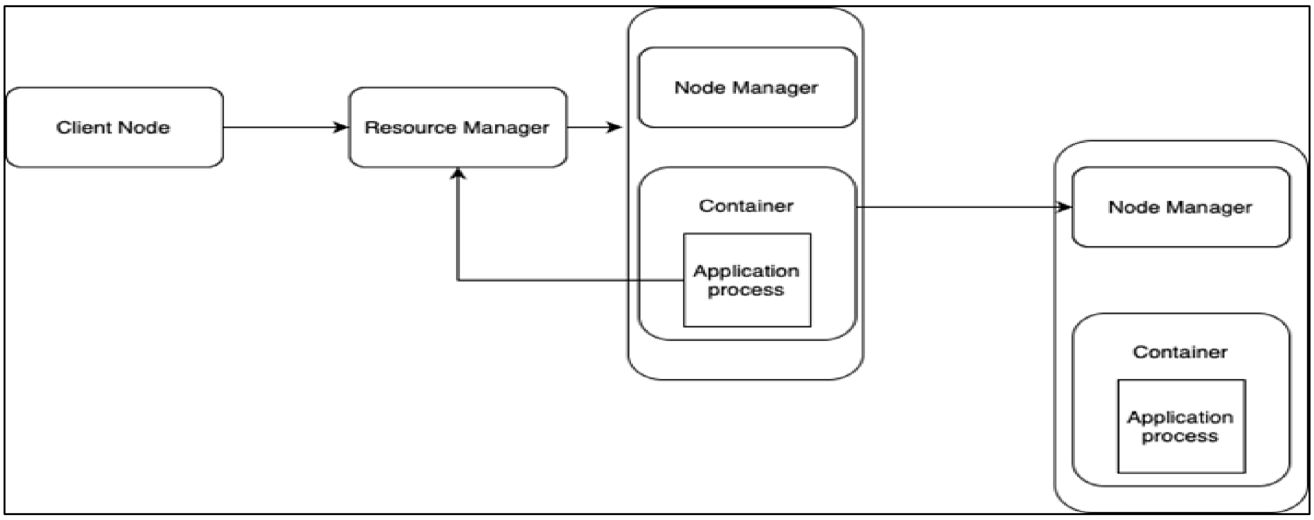

Figure 2. Yarn Architecture

5. Limitations of HDFS and YARN

HDFS and YARN present very good and scalable solutions for distribution storage and resource management but also have issues which must be taken into account. In the case of HDFS a key issue is that we have what in essence is a single point of failure in the Name Node. The Name Node's job is to manage all metadata and it is a very important element for the system to run should it go down the whole cluster goes down at least until failover mechanisms take over. While we have seen the introduction of stand by Name Nodes and failover systems which attempt to address this issue what we see is an increase in the system's complexity and a certain amount of performance hit. Also, in order to ensure fault tolerance, HDFS face issues related to data replication. Although this does improve reliability, a large-scale storage issue, which in very large data sets can lead to poor resource use.

In the case of YARN, we see that resource contention is a significant issue in multi-tenant settings which we see play out when many jobs are competing for the same limited resources. Also we have put in place Fair and Capacity Scheduling which are supposed to even out resource distribution but some jobs still see delay or in some cases don't get any resources at all in dynamic and resource intensive settings. Also YARN has issues with communication between its components which include the Resource Manager, Node Managers, and containers. This in turn introduces latency which in turn affects scheduling performance in large scale clusters. Also, what we find is that current YARN's scheduling policies may not do a great job in heterogeneous environments which have many types of resources (for example CPU and GPU). What we see is that this in turn causes some nodes to be underutilized while others are over used which in the end degrades overall system performance.

6. Performance Evaluation and Simulation Details

In our controlled virtual environment, which we set up using Apache Hadoop framework (v3.3.1), we conducted simulation-based experiments to validate the architectural and scheduling points put forth. We installed the cluster on VirtualBox virtual machines which ran Ubuntu 20.04 with Java 8 and we used the following parameters which we then simulated with the resource management and work load simulation tools of Apache Hadoop YARN:

Number of Nodes: 10 DataNodes, 1 NameNode

Block Size: 128 MB

Memory per Node: 8 GB

Replication Factor: 3

CPU Cores per Node: 4

Tools used: TeraSort, WordCount, and TestDFSIO jobs

Schedulers Used: Fair Scheduler, Capacity Scheduler

Table 2. Quantitative Comparison of Fair and Capacity Schedulers in Hadoop YARN

Metric	Fair Scheduler	Capacity Scheduler
Average Job Completion Time	145 sec	168 sec
Average CPU Utilization	85%	77%
Memory Utilization	83%	84%
TeraSort Throughput	86 MB/s	73 MB/s
Node Failover Recovery	33 sec	36 sec

These results in Table 2 show the issue of resource contention and communication overhead in YARN based systems as well as the trade-off performance between what different scheduling strategies perform. We find that in distributed resource management frameworks' throughput and efficiency plays a major role in architecture selection, this also include block size, replication and scheduler.

This study has analyzed the Fair Scheduler and the Capacity Scheduler based on what they did in terms of resource management and job execution performance in a distributed environment. In many key areas the Fair Scheduler did better. It did a better job at what it does which is run jobs more so for instance it had an Average Job Completion Time of 145 seconds as compared to 168 seconds which the Capacity Scheduler did. Also, it achieved better CPU utilization (85% than the Capacity Scheduler (77%. Which in turn means the Fair Scheduler was more efficient with its compute resources. The Fair Scheduler outperform the Capacity Scheduler in terms of TeraSort Throughput which reported in at 73 MB/s. Also, the Fair Scheduler's Node Failover Recovery time was a little bit faster at 33 seconds as compared to

36. But in terms of memory use the Capacity Scheduler had a little bit of an edge, at 84% of memory use as opposed to Fair Scheduler’s 83% which was a very close second.

7. Conclusion and Future Directions

Distributed resource management is a requirement for the scalability, reliability and performance of present day distributed systems. As these systems grow in scale and complexity, we see that certain issues which still stand out. Scalability is a primary issue we see as we manage thousands of nodes which in turn introduce latency and communication issues. For fault tolerance we require robust recovery which at the same time minimizes redundancy and maintains data availability. In terms of resource contention which is an issue in multi-tenant environments we require balanced approaches for fair allocation and optimal use. Also, the growth of different hardware types which include CPUs, GPUs and specialized accelerators puts forth the need for innovative solutions for compatibility and optimization. Energy efficiency has become a very important issue which we see in the design of algorithms that reduce power use, especially in IoT networks and large-scale data centers.

To that end in the future research, we will consider the adoption of new technology and theories. AI powered systems can be used to enable more real time resource allocation and anomaly detection to improve overall efficiency. Also, we see in edge computing an opportunity for light weight and latency aware resource management which in turn allows for better integration with small scale devices. The serverless architectures will perform effectively in terms of auto scaling and reduction in cold start delays. Also, we see in green computing, a chance to develop energy aware algorithms which in turn promote sustainability. The cross-layer coordination will bring together resource distribution across compute, storage and network layers to enable total system optimization.

References

[1] Zhang, Xiaojie, and Saptarshi Debroy. "Resource management in mobile edge computing: A comprehensive survey." ACM Computing Surveys 55, no. 13s (2023): 1-37.

[2] Huang, Dong, Bingsheng He, and Chunyan Miao. "A survey of resource management in multi-tier web applications." IEEE Communications Surveys & Tutorials 16, no. 3 (2014): 1574-1590.

[3] Moreira, José E., and Vijay K. Naik. "Dynamic resource management on distributed systems using reconfigurable applications." IBM Journal of Research and Development 41, no. 3 (1997): 303-330.

[4] Hussain, Hameed, Saif Ur Rehman Malik, Abdul Hameed, Samee Ullah Khan, Gage Bickler, Nasro Min-Allah, Muhammad Bilal Qureshi et al. "A survey on resource allocation in high performance distributed computing systems." Parallel Computing 39, no. 11 (2013): 709-736.

[5] Zahoor, Saniya, and Roohie Naaz Mir. "Resource management in pervasive Internet of Things: A survey." Journal of King Saud University-Computer and Information Sciences 33, no. 8 (2021): 921-935.

[6] Cheng, D., Rao, J., Jiang, C., & Zhou, X. (2015, May). Resource and deadline-aware job scheduling in dynamic hadoop clusters. In 2015 IEEE International Parallel and Distributed Processing Symposium. IEEE: 956-965.

[7] White, T. (2012). Hadoop: The definitive guide. " O'Reilly Media, Inc.".

[8] Krauter, Klaus, Rajkumar Buyya, and Muthucumaru Maheswaran. "A taxonomy and survey of grid resource management systems for distributed computing." Software: Practice and Experience 32, no. 2 (2002): 135-164.

[9] Borthakur, Dhruba, Jonathan Gray, Joydeep Sen Sarma, Kannan Muthukkaruppan, Nicolas Spiegelberg, Hairong Kuang, Karthik Ranganathan et al. "Apache hadoop goes realtime at facebook." In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 1071-1080. 2011.

[10] Moreira, José E., and Vijay K. Naik. "Dynamic resource management on distributed systems using reconfigurable applications." IBM Journal of Research and Development 41, no. 3 (1997): 303-330.

[11] Chen, Chen-Chun, Kai-Siang Wang, Yu-Tung Hsiao, and Jerry Chou. "ALBERT: an automatic learning based execution and resource management system for optimizing Hadoop workload in clouds." Journal of Parallel and Distributed Computing 168 (2022): 45-56.

[12] Huang, Dan, Jun Wang, Qing Liu, Nong Xiao, Huafeng Wu, and Jiangling Yin. "Enhancing proportional IO sharing on containerized big data file systems." IEEE Transactions on Computers 70, no. 12 (2020): 2083-2097.

[13] Rao, B. Thirumala, and L. S. S. Reddy. "Survey on improved scheduling in Hadoop MapReduce in cloud environments." arXiv preprint arXiv:1207.0780 (2012).

[14] Yao, Yi, Han Gao, Jiayin Wang, Bo Sheng, and Ningfang Mi. "New scheduling algorithms for improving performance and resource utilization in hadoop YARN clusters." IEEE Transactions on Cloud Computing 9, no. 3 (2019): 1158-1171.

[15] Van Do, Tien, Binh T. Vu, Nam H. Do, Lóránt Farkas, Csaba Rotter, and Tamás Tarjányi. "Building block components to control a data rate in the Apache Hadoop compute platform." In 2015 18th International Conference on Intelligence in Next Generation Networks, IEEE, (2015): 23-29