translab.io

# BIG DATA ON FUNGIBLE COMPOSABLE INFRASTRUCTURE WHITEPAPER

# TABLE OF CONTENTS

# 1 EXECUTIVE SUMMARY

Unlike most IT applications, Big Data applications are tightly coupled with business transformations and goals. Hence, performance becomes a key parameter as businesses need to extract real-time insights from Hadoop clusters.

Traditional clusters built on commodity x86 hardware face severe performance bottlenecks as general-purpose CPUs are tasked with running data-centric jobs. This not only hampers the Big Data jobs by deallocating CPU cores, but is also inefficient in handling those data-centric jobs.

With the Fungible Data Processing Unit (DPU) powering both the Fungible TrueFabric and Fungible Composer, business teams and IT teams are freed from such limitations and inefficiencies of traditional clusters. The Fungible Data Center (FDC) is a preconfigured rack-scale appliance that offers performance at or near theoretical limits. Hadoop operators can create clusters on the fly taking advantage of the composable nature of the FDC; eliminating overprovisioning and providing the elasticity needed for lean, agile organisations of today.

Through our intensive benchmarking, we found that the Fungible Data Center solution is a no-compromise platform for Big Data and Hadoop, offering almost no performance reduction with increasing resiliency from RF2 to RF3.

# 2 INTRODUCTION TO BIG DATA AND HADOOP

Though the term “Big Data” dates way back to the 1960s, the concept did not become mainstream until the rise of the World Wide Web and emergence of platforms like Yahoo!, Google, and Facebook, in the 1990s and 2000s. These platforms enabled the growth of unstructured data on a petabyte scale. **Managing and monetizing** these large datasets became the greatest tech challenge of all.

The solution to this challenge came in the form of two Google research papers: **Google distributed File System (GFS)**, which described a distributed architecture for storing large datasets, and **MapReduce**, which described a parallel processing model to handle large datasets.

These two research papers were realised as the **Apache Hadoop Project.** Today, Hadoop forms the foundation of Big Data architecture.

## 2.1 Hadoop

Apache Hadoop Project designed a **novel distributed architecture** based on commodity x86 architecture instead of traditional "Big Iron". The architecture was designed to be fault-tolerant rather than fault-resistant. The principle was to recover from any fault as quickly as possible by creating multiple copies of each file. The number of replicas of data is determined based on resiliency required (**Tuneable Replication**).

Another unique aspect of the architecture was to **Move the Compute closer to the Data**. Due to limitations of networking, the trade-off was made to minimize network congestion by minimizing the bandwidth needed for data replication. To reduce network congestion, the compute and the storage were combined into a single unit. However, this necessitated compute handle the storage workloads, which were traditionally offloaded to dedicated controllers.

## 2.2 Hadoop Subsystems

Hadoop consists of 3 core subsystems:

- HDFS (Hadoop Distributed File System)
- YARN (Yet Another Resource Negotiator)
- MapReduce

### 2.2.1 HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

HDFS is the storage subsystem of Hadoop. To make the overall system fault-tolerant, three separate copies of data are stored across multiple nodes and server racks. This ensures that in case of a node or even an entire rack failure, no data is lost. HDFS splits large datasets into smaller chunks called blocks. These blocks are then distributed across the entire HDFS cluster.

HDFS has a typical master-slave architecture, with one master node (called **NameNode)** managing multiple slave nodes (called **DataNodes**) within the Hadoop cluster. NameNode, the most vital component of a Hadoop cluster, stores the metadata (i.e, file names, their permissions, IDs, locations, as well as the number of replicas). Loss of NameNode will lead to loss of data across the Hadoop cluster. Additional NameNode is available in active-passive (**Secondary NameNode**) or active-active (**Standby NameNode**) mode to ensure there are no single points of failure.

### 2.2.2 YET ANOTHER RESOURCE NEGOTIATOR (YARN)

YARN manages all the resources in a Hadoop cluster. It, in turn, has four components: Resource Manager, Node Manager, Application Master and Container.

Resource Manager, which runs on the NameNode, accepts job submissions, and allocates resources to those jobs.

What the Resource Manager is to NameNode, Node Manager is to DataNode. Each DataNode runs a separate instance of Node Manager. After initiation, Node Manager registers itself with the Resource Manager and manages the application containers assigned to it by the Resource Manager.

When a job is submitted to the Resource Manager, it assigns the job to an Application Master, based on available resources. The Container is a logical representation of physical resources on a single DataNode.

### 2.2.3 MAPREDUCE

MapReduce processes large datasets stored on HDFS. MapReduce runs on the same DataNode where the data is stored (moving the compute closer to storage as mentioned earlier). The MapReduce algorithm then maps, shifts, sorts, and reduces the data to provide the desired results. This output is stored in HDFS.

# 3 CLOUDERA DATA PLATFORM

In 2008, Christophe Bisciglia, Amr Awadallah and Jeff Hammerbacher, pioneers and proselytisers in the field of Big Data, founded Cloudera. At Cloudera, they developed Cloudera's Distribution including Apache Hadoop (CDH). CDH was not a single tool, but a combination of various Big Data tools, like Apache Hadoop, Apache Spark, Apache Flume, Apache Impala, Apache Kudu and Apache HBase, needed to provide a comprehensive system to build a Big Data Lake. In 2019, Cloudera acquired its competitor, Hortonworks, and combined CDH with Hortonworks Data Platform (HDP) into a single platform: **Cloudera Data Platform (CDP)**.

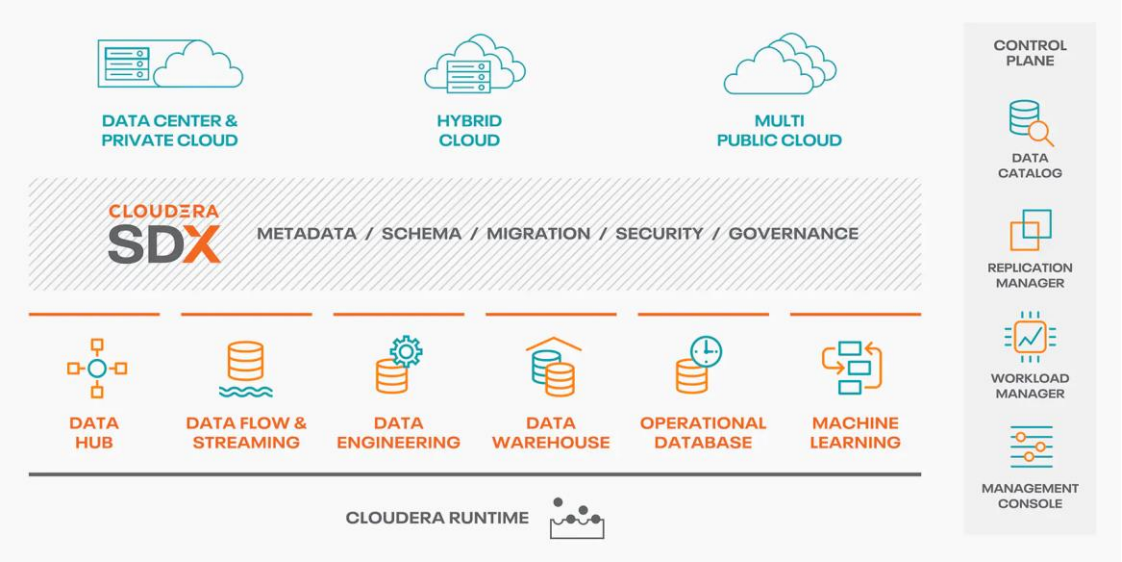


*Figure 1 Cloudera Data Platform*

Instead of deprecating HDP, Cloudera took the best tools from both CDH and HDP and updated them into a single platform. The result was a microservices-oriented, cloud-native data lake platform with advanced analytics, centralized management, and built-in security and governance.

Cloudera is available on-prem as CDP Private Cloud or as CDP Public Cloud.

# 4 CHALLENGES WITH HADOOP ARCHITECTURE

Combining compute and storage into a single node is one of the key aspects of Hadoop architecture. This was meant to address all performance and management issues with distributed systems for handling large datasets. However, limitations soon became very apparent.

To scale storage, one must scale compute too. Reverse is also true. This has implications with respect to software licenses as each additional node must be licensed.

In a traditional architecture with centralized storage, there are dedicated storage controllers to take care of data tasks, like encryption, deduplication, compression, etc. These controllers not only offload these functions from the CPU but also are purpose-built for such data tasks. General-purpose CPUs are not efficient in handling such tasks. Further, when a CPU core is tasked with handling storage functions, there is one less core available to handle Big Data functions. This reduces the overall performance of the system.

Finally, the architecture is designed to run the compute workload on the same node as where the data is located (data localization). This leads to reduction in capacity utilization as storage cannot be pooled.

# 5 COMPOSABLE INFRASTRUCTURE

The rise of Hadoop and other distributed applications led to a shift from traditional **three-tier infrastructure** (where compute and storage were segregated and were connected via network and/or SAN switches) to **hyperconverged infrastructure** (where the storage is locally attached to the compute).

Hyperconverged infrastructure (HCI) followed a Hadoop-like principle by bringing compute and storage on the same HCI node and ensuring fault-tolerance through having multiple replicas of data. It aimed to solve problems IT departments were facing: "**Flash storage performance was much lower than its theoretical limits"**. By attaching flash storage directly with the compute through a PCIe bus, HCI aimed to provide IOPS and latency close to theoretical limits.

However, HCI faced the same challenge as mentioned above:

- Unable to scale compute and storage independently
- Lower compute performance due to storage overhead
- Reduced capacity utilization

**Composable Infrastructure** resolves the above challenges by offloading the storage services from CPU to a dedicated storage controller called a **Data Processing Unit** or DPU. In Composable Infrastructure, storage services can now be offloaded from CPUs to DPUs, leading to increased compute and storage performance. DPUs can provide storage services, such as compression, deduplication, and erasure coding, **in-line and at-line rate**. This creates a highly scalable, low-latency disaggregated storage that can scale independently of compute and has **full separation between control plane and data plane**.

# 6 FUNGIBLE, INC.

Fungible was founded in 2015 with a vision of providing "solutions that will enable highly performant, efficient, reliable and secure data centers to be built at any scale." In 2016, Fungible developed its first Data Processing Unit (DPU), which forms the core of Fungible's dynamically composable infrastructure offerings.

## 6.1 Fungible DPU

The Fungible DPU is the so-called "third socket", with CPU and GPU being the first two. The CPU is the general-purpose processor handling all types of tasks, while the GPU is a specialised processor handling vector floating point operations, typically graphical or AI/ML workloads.

The Fungible DPU is a system-on-a-chip (SOC) combining a general purpose multi-threaded processor with Ethernet and PCIe interfaces. It runs a custom operating system, FungibleOS, designed for data-centric tasks. The DPU is highly-programmable with the ability to make changes to protocols quickly and without performance degradation.

The DPU can be deployed either inside an application server, where the DPU acts as an endpoint device, or inside a storage server, where it replaces the CPU. In an application server, the Fungible DPU mediates between the CPU and network, offloading data-centric computations from the CPU. In a storage server, the Fungible DPU implements the target-side functionality of a storage system. Thus, the Fungible DPU efficiently disaggregates both compute and storage resources.

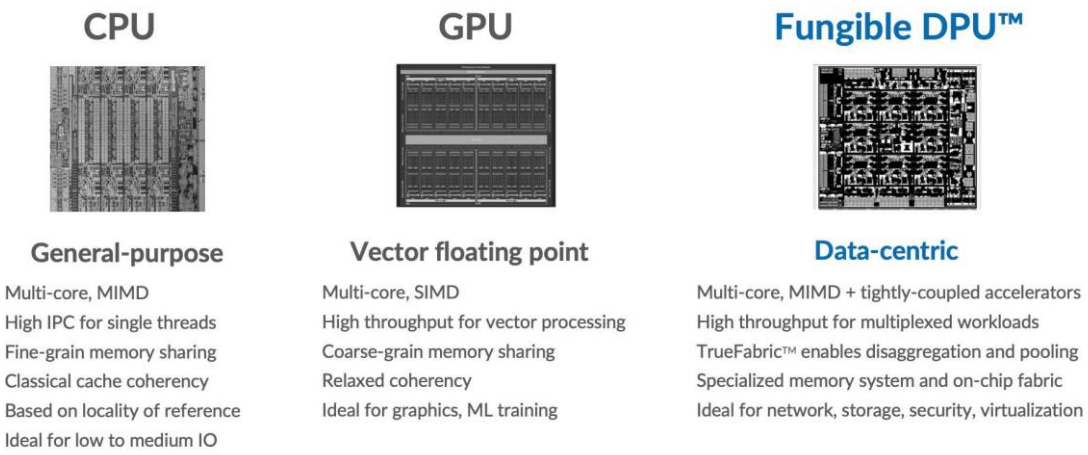


*Figure 2 Three Sockets (Source: The Fungible DPU: New Category of Microprocessor)*

## 6.2 Fungible Architecture

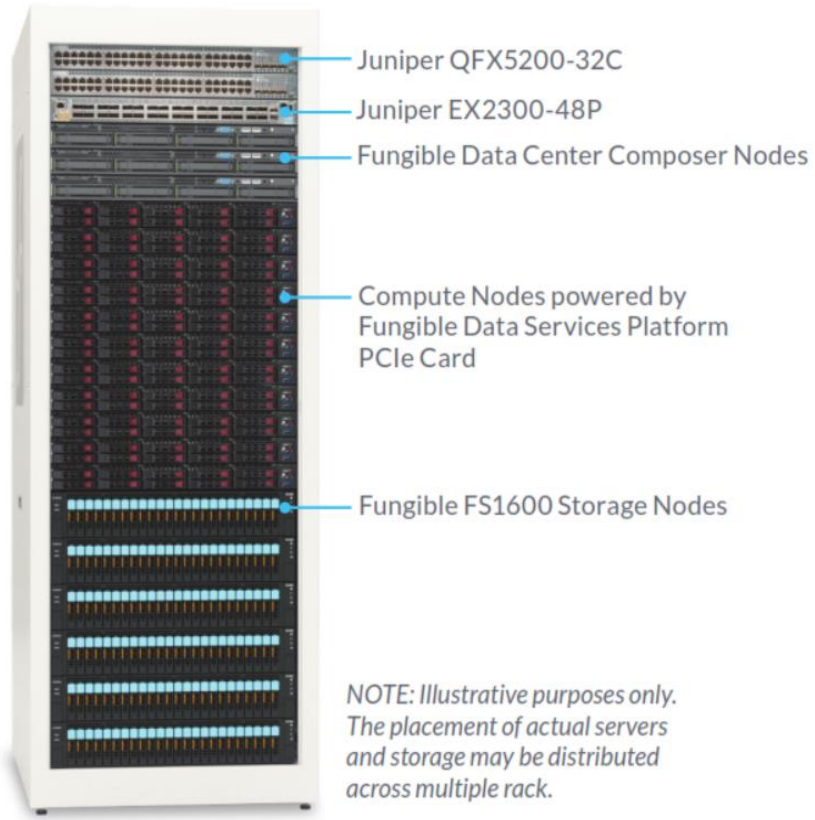


*Figure 3 Typical FDC rack*

The **Fungible Solution** represents a data center-scale implementation of Fungible's vision. It is an on-prem, turn-key offering that consists of racks of disaggregated application and storage servers. The entire FDC is managed by Fungible Composer software.

The key components of the solution are:

- Two or more high-performance, scale-out NVMe-oF **Fungible FS1600 nodes**
- Diskless **Application servers** with x86 CPU and one Fungible Accelerator Card
- **Three dedicated management** nodes that host the Fungible Composer Software
- **Top-of-Rack (ToR) switches** for data and management networks and routers from Juniper

Fungible Solution is available in five storage configurations and four compute configurations. Users can choose any one of the 20 compute and storage combinations that meet their needs.

*Table 1 Fungible T-shirt Sizes*

| | Compute.Tiny | Compute.Medium | Compute.Large | Compute.XLarge |
|---|---|---|---|---|
| Storage.Tiny | (2) FS1600, 92TB<br>(4) Compute Nodes, 2*8x3.1Ghz, 256GB RAM<br>Composer nodes | (2) FS1600, 92TB<br>(8) Compute Nodes, 2*8x3.1Ghz, 512GB RAM<br>Composer nodes | (2) FS1600, 92TB<br>(32) Compute Nodes, 2*12x2.1Ghz, 512GB RAM<br>Composer nodes | (2) FS1600, 92TB<br>(32) Compute Nodes, 2*20x2.1Ghz, 1.5TB RAM<br>Composer nodes |
| Storage.Small | (2) FS1600, 92TB (4) Compute Nodes, 2*8x3.1Ghz, 256GB RAM<br>Composer nodes | (2) FS1600, 92TB<br>(8) Compute Nodes, 2*8x3.1Ghz, 512GB RAM<br>Composer nodes | (2) FS1600, 92TB<br>(32) Compute Nodes, 2*12x2.1Ghz, 512GB RAM<br>Composer nodes | (2) FS1600, 92TB<br>(32) Compute Nodes, 2*20x2.1Ghz, 1.5TB RAM<br>Composer nodes |
| Storage.Medium | (3) FS1600, 138TB<br>(4) Compute Nodes, 2*8x3.1Ghz, 256GB RAM<br>Composer nodes | (3) FS1600, 138TB<br>(8) Compute Nodes, 2*8x3.1Ghz, 512GB RAM<br>Composer nodes | (3) FS1600, 138TB<br>(32) Compute Nodes, 2*12x2.1Ghz, 512GB RAM<br>Composer nodes | (3) FS1600, 138TB<br>(32) Compute Nodes, 2*20x2.1Ghz, 1.5TB RAM<br>Composer nodes |
| Storage.Large.Perf | (7) FS1600, 323TB<br>(4) Compute Nodes, 2*8x3.1Ghz, 256GB RAM<br>Composer nodes | (7) FS1600, 323TB<br>(8) Compute Nodes, 2*8x3.1Ghz, 512GB RAM<br>Composer nodes | (7) FS1600, 323TB<br>(32) Compute Nodes, 2*12x2.1Ghz, 512GB RAM<br>Composer nodes | (7) FS1600, 323TB<br>(32) Compute Nodes, 2*20x2.1Ghz, 1.5TB RAM<br>Composer nodes |
| Storage.Large.Density | (7) FS1600, 1290TB<br>(4) Compute Nodes, 2*8x3.1Ghz, 256GB RAM<br>Composer nodes | (7) FS1600, 1290TB<br>(8) Compute Nodes, 2*8x3.1Ghz, 512GB RAM<br>Composer nodes | (7) FS1600, 1290TB<br>(32) Compute Nodes, 2*12x2.1Ghz, 512GB RAM<br>Composer nodes | (7) FS1600, 1290TB<br>(32) Compute Nodes, 2*20x2.1Ghz, 1.5TB RAM<br>Composer nodes |

# 7 ADDRESSING THE BIG DATA CHALLENGES

The Fungible solution is DPU powered and resolves the issues present in Hadoop architecture.

First, it disaggregates the compute and storage into separate application and storage nodes. This means increasing storage capacity has no impact on software licensing. This is evident from the FDC T-shirt sizes as shown in table above.

The DPU is built from the ground up to handle storage functions. Both the data path and the control plane of the DPU are fully programmable using high-level programming languages. By combining data-centric accelerators with processor cores, Fungible DPUs can provide exceptional performance without relying on CPUs. The CPUs can exclusively handle Big Data workloads. Thus, Fungible can provide near-theoretical performance on NVMe storage.

As data is not localised, but available over networking fabric, multiple storage capacities can be pooled together into fewer nodes leading to higher capacity utilization.

# 8 BENCHMARKS

## 8.1 Test Bed Configuration

The Hadoop cluster was deployed on the Fungible Solution. Below is the detailed configuration:

- Nodes – CPU – 16 cores – RAM – 128 GB
- Storage – 4 *1000GB, 4*500GB, 4*100GB volumes
- Network – 2*50Gbps network connectivity between nodes.
- Operating System - Ubuntu 20.4
- Hadoop - Cloudera-7.1.7, Hadoop- 3.2

## 8.2 Cloudera Components Distribution

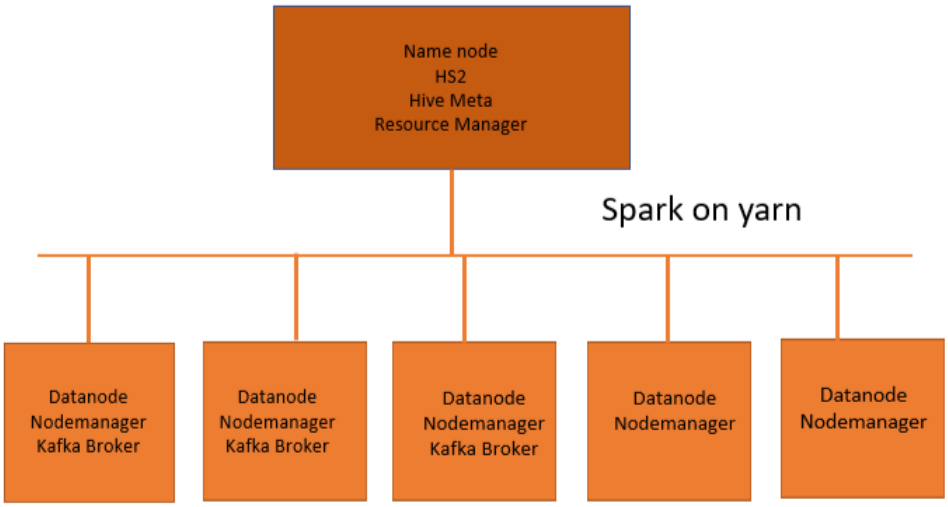


Figure 4 Cloudera Component Distribution

## 8.3 Hadoop Benchmark

To benchmark the Fungible, we ran the TeraSort benchmark, which gives testers an estimate of Hadoop storage and MapReduce performance.

TeraSort benchmark run is a three-step process:

1. Generating the input data via TeraGen.
2. Running the actual TeraSort on the input data.

3. Validating the sorted output data via TeraValidate.

### 8.3.1 HOW TO RUN TERAGEN, TERASORT AND TERAVALIDATE

Before performing the performance test, you should review the folder hadoop_examples.jar

**Running TeraGen**

Run this command:

```
hadoop jar hadoop_examples.jar TerraGen 100000000 /benchmarks/terasort-input // 10GB
```

This creates a 500GB file in the HDFS under the /benchmarks/terasort-input folder. It enables the TeraSort to run its benchmark.

**Running TeraSort**

Run this command:

```
hadoop jar hadoop_examples.jar terasort /benchmarks/terasort-input /benchmarks/terasort-output
```

This command runs a benchmarking MapReduce job on the data created by TeraGen in the /benchmarks/terasort-input folder. The output result of the reduction is written into files and placed under the /benchmarks/terasort-output/ folder, where TeraValidate will check later.

**Running TeraValidate**

Run this command:

```
hadoop jar hadoop_examples.jar teravalidate /benchmarks/terasort-output /benchmarks/terasort-validate
```

This command just ensures if the output of TeraSort was valid and without error.

**Hadoop Configuration**

| SN | Configuration item | Final tuning |
|---|---|---|
| 1 | dfs.namenode.handler.count | 160 |
| 2 | fs.trash.interval | default |
| 3 | io.file.buffer.size | default |
| 4 | yarn.nodemanager.resource.memory-mb (All nodes: 50*5) | 250GB |
| 5 | yarn.nodemanager.resource.cpu-vcores(Average) | 50 |
| 6 | yarn_scheduler_minimum_allocation_mb | 1GB |

| | | |
|---|---|---|
| 7 | yarn.scheduler.maximum-allocation-mb | 32GB |
| 8 | mapreduce.map.memory.mb | 2GB |
| 9 | mapreduce.reduce.memory.mb | 4GB |
| 10 | mapreduce.map.java.opts | defaults |
| 11 | mapreduce.reduce.java.opts | defaults |
| 12 | mapred_compress_map_output | TRUE |
| 13 | mapred_map_output_compression_codec | snappy |
| 14 | mapred_reduce_parallel_copies | 32 |
| 15 | mapreduce.task.io.sort.mb | 512 |
| 16 | mapreduce.map.sort.spill.percent | 0.8 |
| 17 | mapreduce.task.io.sort.factor | 64 |

### 8.3.2 RESULTS

Benchmark was run with RF2 and RF3 configuration with 100 GB data in both cases.

*Table 2 RF3 Benchmark Results*

| | Data Size | RF | Jobs | Time |
|---|---|---|---|---|
| **TeraGen** | 100GB | 3 | 100 | 156 sec |
| **TeraSort** | 100GB | 3 | 100 | 743 sec |
| **TeraValidate** | 100GB | 3 | 100 | 84 sec |

*Table 3 RF2 Benchmark Results*

| | Data Size | RF | Jobs | Time |
|---|---|---|---|---|
| **TeraGen** | 100GB | 2 | 100 | 153 sec |
| **TeraSort** | 100GB | 2 | 100 | 744 sec |
| **TeraValidate** | 100GB | 2 | 100 | 77 sec |

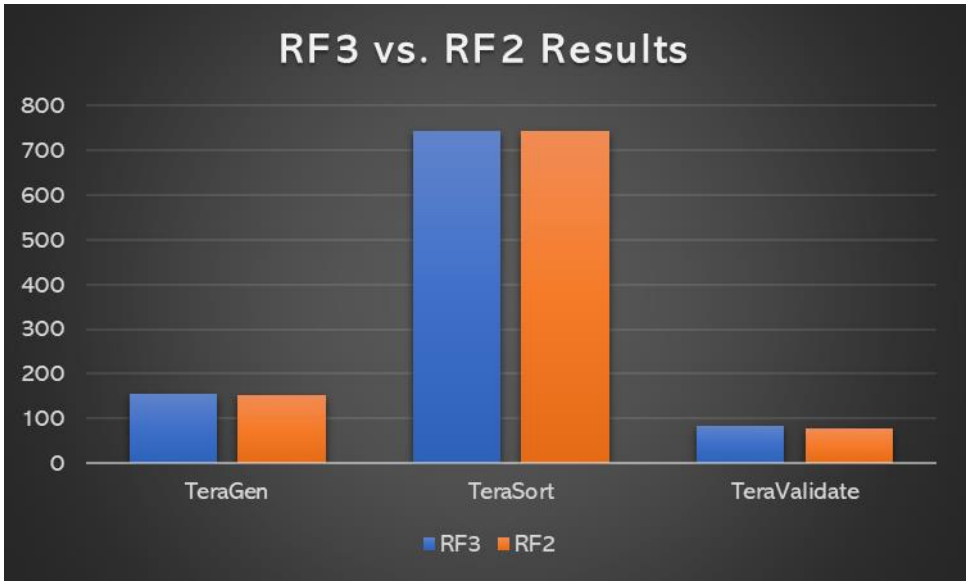


*Figure 5 Benchmark Comparison*

# 9 FINAL THOUGHTS

Hadoop operators have always faced a trade-off with respect to performance and resiliency. The distributed architecture of Hadoop meant there was no inherent data protection mechanism like RAID. Instead, data protection was enabled by storing multiple copies of data. With RF 2, two copies of data are maintained (akin to RAID 1). With RF3, three copies of data are maintained. While RF3 gave better resiliency, the overheads associated with it led to lower performance and higher cost of the Hadoop cluster. As a result, Hadoop operators reserved RF3 for extremely critical production data.

With the Fungible Solution powered by the Fungible DPU, Hadoop operators do not have to make such trade-offs. As evident from the benchmark results, having an additional copy of data in the Hadoop cluster did not have any noticeable effect on cluster performance.

Overall, considering other benefits of Fungible Data Center (like, storage offload, better resource pooling, etc), it is the right platform for any Hadoop workload.

# 10 AUTHORS

Gaganjit Singh Wazir, CEO at Translab Technologies
Neeraj G Dhopte, CTO at Translab Technologies
Shirshendu Bikash Mandal, VP, Technology Solutions at Translab Technologies
Anuj Kumar Jain, Technology Specialist at Translab Technologies

# 11 ABOUT TRANSLAB

Translab Technologies enables Digital Transformation for enterprises worldwide by providing seamless customer experience, business agility and actionable insights. Utilizing disruptive technologies, like Big Data, Analytics, the Internet of Things, Automation, Mobility and Cloud, we offer solutions that have applications across industry sectors. Translab Technologies is headquartered in Bengaluru and provides support to its customers worldwide. Translab has designed solutions for data-warehousing and Bigdata analytics for customers in BFSI, Insurance & healthcare to provide business insights on large volumes of data that yield results for various use cases. Some of the key use cases are Customer360, Sentimental Behaviour Analytics, and Descriptive Analytics etc. Translab Technologies has been Oracle Gold Partner with deep expertise in Engineered systems, Private Cloud Appliance, databases, and Analytics Tools. Translab has also partnered with Cloudera to address big data challenges and has deep expertise in designing and deploying Big Data clusters using Cloudera technology.