Of Mice and Elephants

DBS | SRE CON 2021

DBS

World’s Best Yet Again

World’s Best Bank 2021, Euromoney
Best Bank in the World 2020, Global Finance
World’s Best Bank 2019, Euromoney
Global Bank of the Year 2018, The Banker

Lim Koon Seng

Koon Seng received a Masters in Electrical Engineering from Columbia University in 1996 and a Bachelors in Computer Science from NUS in 1991. By day, he is an Executive Director and heads the SRE team of DBS Middle Office.

By night, he prowls through code with his trusty pet snake Python 3, hunting for the occasional bug. In his past life, he spent 17 years founding and working for various startups in the US before returning to Singapore in 2010.

Sandeep Hooda

Sandeep leads the SRE team focused on learning from the incident, containerisation, and responsible for various SDLC tools specialising in DevOps.

Prior to this, he was managing cloud infrastructure teams where his responsibilities were usually automation, infrastructure architecture, and working closely with solution architects.

He is a passionate user of open source with a strong focus on creating quintessential solutions. He conducts workshops to educate and spread awareness on blameless culture "from tech incidents to biz decisions."
He is a Sci-Fi lover, with a keen interest in astronomy, dreams of space exploration, and sailing around the world.

Speaker Introduction

Diving Right In…

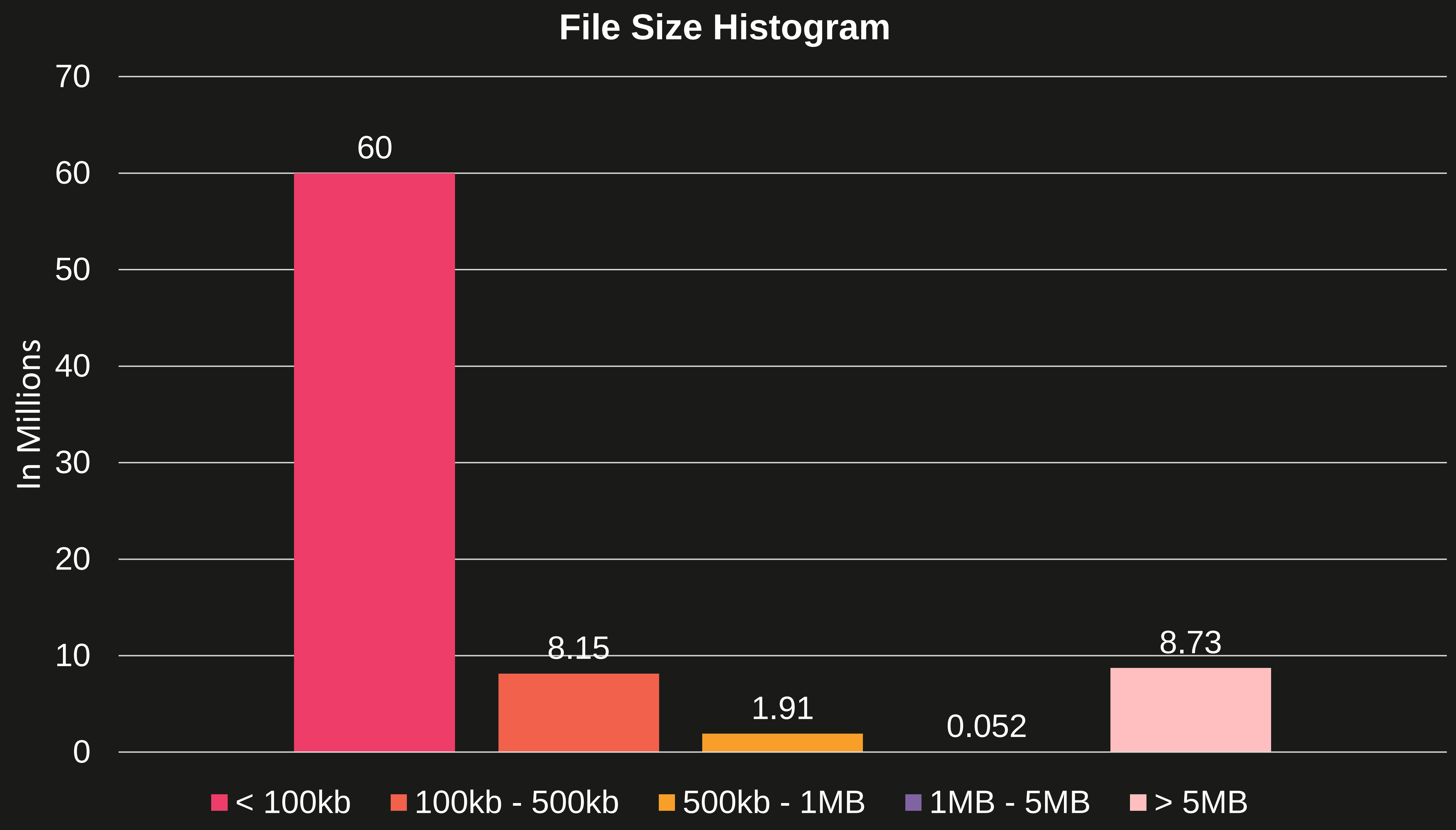
Our Fast-growing Data Lake (circa 2017-2019)

- Cluster cores grew x3 from 720 to 2.5K cores
- Cluster memory grew x3 from 7.6TB to 22TB
- Cluster storage grew x2.5 from 660TB to 888TB
- Node local disk storage supplemented by virtualised file system backed by remote object store
- Physical nodes supplemented by virtualised data nodes

But in the shadows…

… A silent menace lurked

88% of ALL files were less than 5MB and were scattered all over the node’s local disk and virtualised file system.



One fateful dark and stormy night on May 31st 2019…

- Software bug in test job which led it to write arbitrarily long paths to virtual filesystem, resulting in 2,900 failures per minute
- Firmware bug in object store appliance triggers repeated reboots
- Cluster senses data node filesystem reboot triggers read-scanning to repair
- Millions of small files slow down read-scanning and reduces effective throughput of appliance to 30 – 90 MB per second
- **Cluster recovery stuck in limbo, waiting for read scan to complete before HDFS can come up**

The aftermath and…

A blameless recovery!

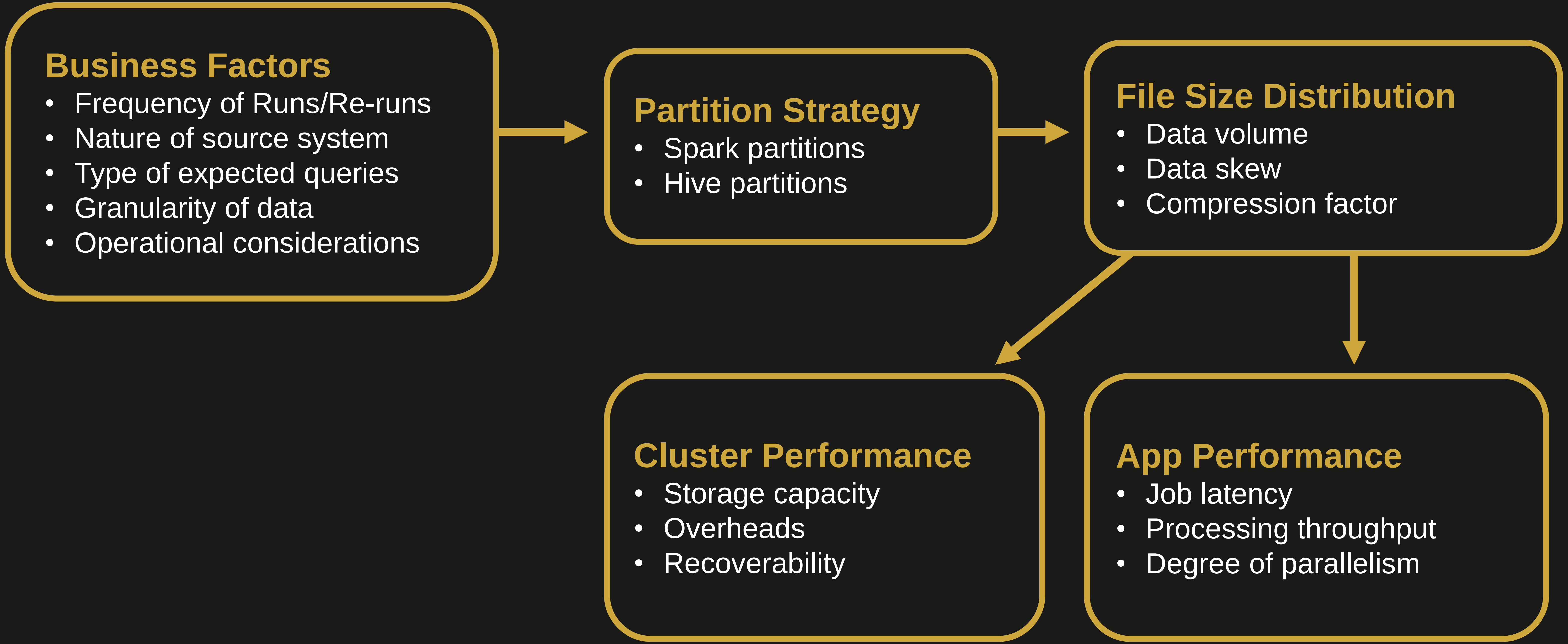
► **Approach**

- Copy out blocks from object store to local disks & SAN to overcome network & seek latency of object store
- Manually recover HDFS, partition by partition, file by file
- Prioritise recovery by age of data & criticality of app

► **Took a week to complete full recovery**

► **Focused on lessons learnt and improvements, not on blame**

Origins and Impact of Small Files



Business Factors

- Frequency of Runs/Re-runs
- Nature of source system
- Type of expected queries
- Granularity of data
- Operational considerations

Partition Strategy

- Spark partitions
- Hive partitions

File Size Distribution

- Data volume
- Data skew
- Compression factor

Cluster Performance

- Storage capacity
- Overheads
- Recoverability

App Performance

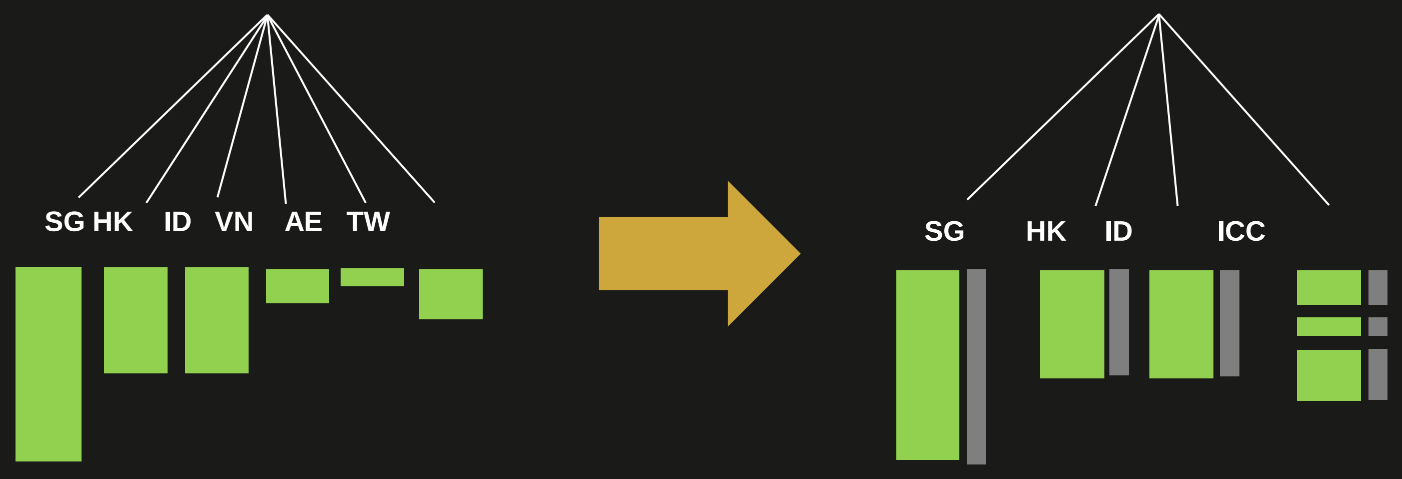
- Job latency
- Processing throughput
- Degree of parallelism

♦ Hive Partitions VS ♦ Spark Partitions

♦ Hive Partitions

(on HDFS)

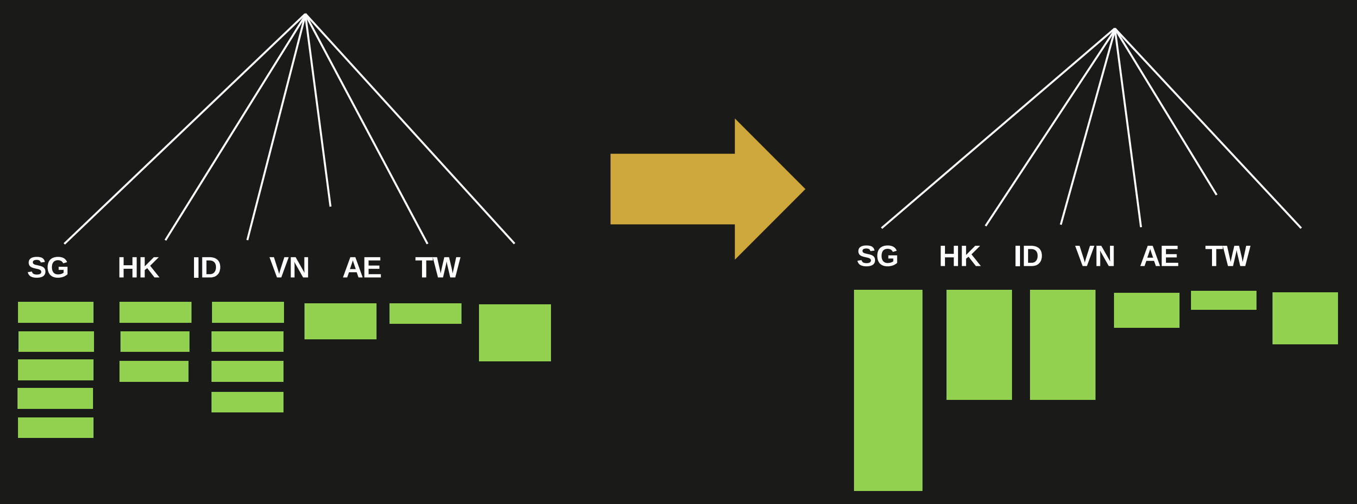
- Split data so that columns with similar value are in same directory
- Reduces overhead of loading a large file only to select a small subset of data
- Increases number of partitions and higher overheads compiling Hive & Impala queries
- Potentially fragments data into directories with small files



♦ Spark Partitions

(in memory)

- Split data into smaller chunks on read so each can be processed by a separate executor
- Possible because parquet file format allows partial read of file by an executor
- Increases parallelism of stages so overall batch completes faster
- Increases speed by reducing memory footprint per executor
- Potentially fragments data into small compressed files when saving back



Overheads of Small Files

1 HDFS block can store up to 128MB, *M,* and incurs 150 bytes heap memory, *B,* on name node. Assuming a cluster uses 3 x replication, *R,*

Given a 1GB, T, dataset is broken into

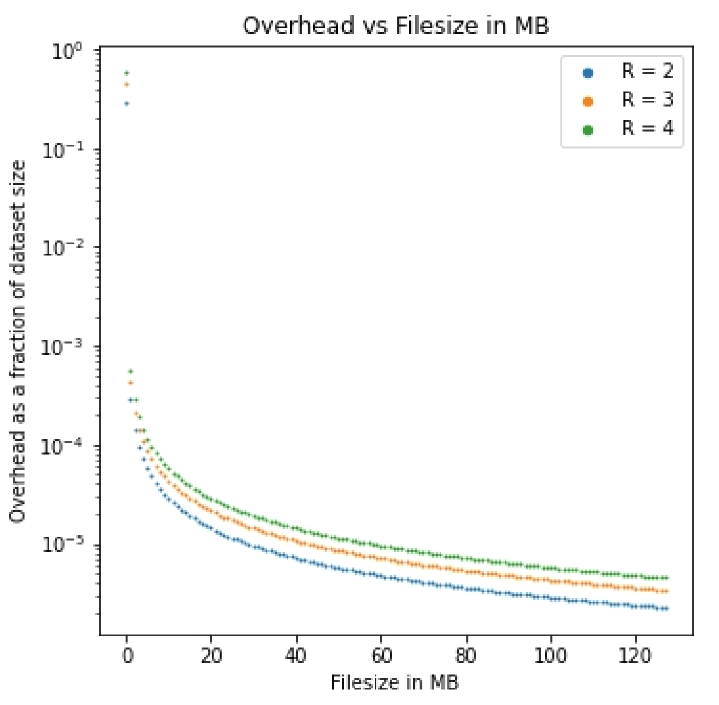
- Files, *F*, of 128MB, heap requirement = ((1024/128)*3+1)*150 = 3.7K ≈ **0.0003%** overhead
- Files, *F*, of 1MB, heap requirement = ((1024/1)*3+1)*150 = 460K ≈ **0.042%** overhead
- Files, *F*, of 1kb, heap requirement = ((1024/(1/1024)*3+1)*150 = 471MB ≈ **45%** overhead

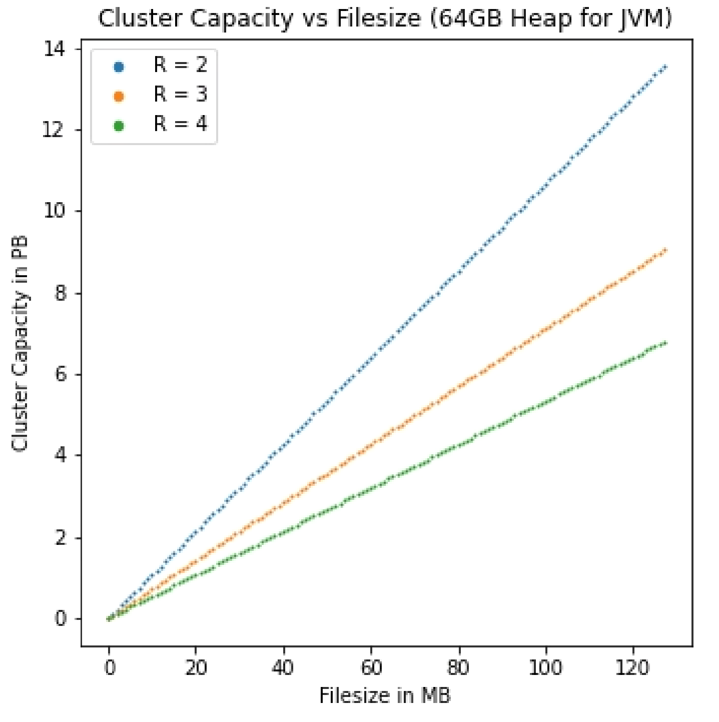
Overhead (relative to dataset size) = $\left(\left(\frac{T}{\min(M,F)}\right)R+1\right)\frac{B}{T} \approx \frac{RB}{F} \approx \frac{150R}{F}$

Given heap size of H, max # of files in cluster = $\frac{H}{2BR}$ → max cluster capacity $\geq \frac{HF}{2BR}$

Heap memory overhead increases exponentially with decreasing file size.

Cluster capacity & max file counts reduces linearly with decreasing file size.





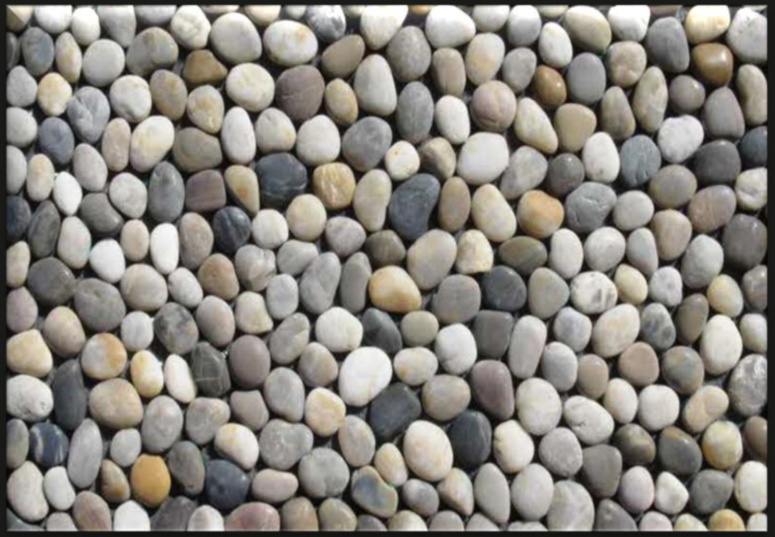
SRE-ious Retrospective

SRE Way of Deep Diving

Workflows	**Ingestion** ✓ Sqoop ✓ Spark ✓ Kafka ✓ Map-Reduce ✓ Nifi	**Transform** ✓ Spark ✓ Map-Reduce
Small file contributors	• *Input files from sources are small* • *Ingestion from Kafka* • *Poor tuning practices during Ingestion* • *No clear understanding of Bigdata tools* • *Best Practices not followed*	• *Application framework such as Spark* • *Poor tuning practices for partitions during transformation* • *No clear understanding of Bigdata tools* • *Best Practices not followed*
Actions by Dev Teams	✓ ***Delete** the source files.* ✓ ***Archive** the source files.*	✓ ***Merge** the hdfs small files after “X” number of days* ✓ ***Purge** the data based on business policy* ✓ ***Copy** the HDFS files to S3 bucket or other sources*

This is not sustainable and not SRE...

SRE Approach to Dimension the Problem

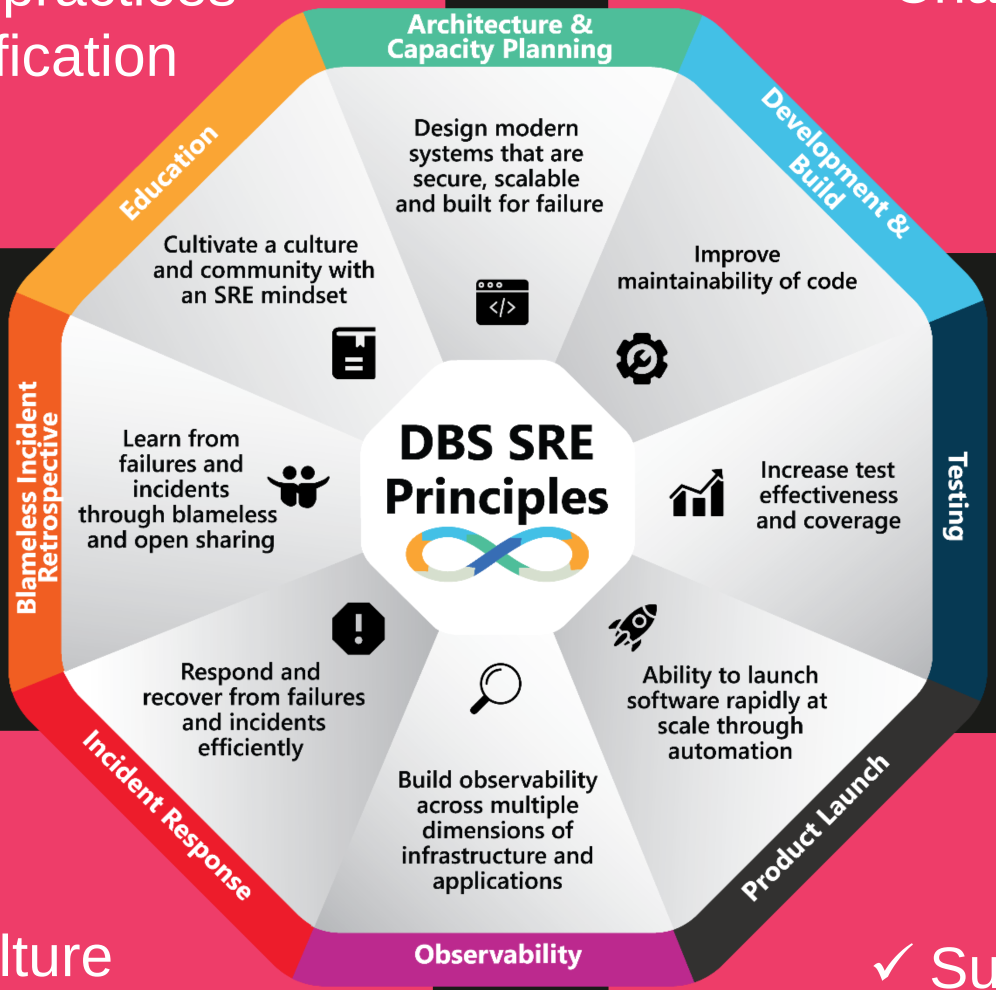


Learning

- ✓ Awareness
- ✓ Best practices
- ✓ Certification

Deep Engineering

- ✓ Architecture
- ✓ Chaos Engineering



DBS SRE Principles

Architecture & Capacity Planning

Design modern systems that are secure, scalable and built for failure

Development & Build

Improve maintainability of code

Testing

Increase test effectiveness and coverage

Product Launch

Ability to launch software rapidly at scale through automation

Observability

Build observability across multiple dimensions of infrastructure and applications

Incident Response

Respond and recover from failures and incidents efficiently

Blameless Incident Retrospective

Learn from failures and incidents through blameless and open sharing

Education

Cultivate a culture and community with an SRE mindset

- ✓ Blameless culture
- ✓ Engagement

Collaboration

- ✓ Sustainability
- ✓ Observability

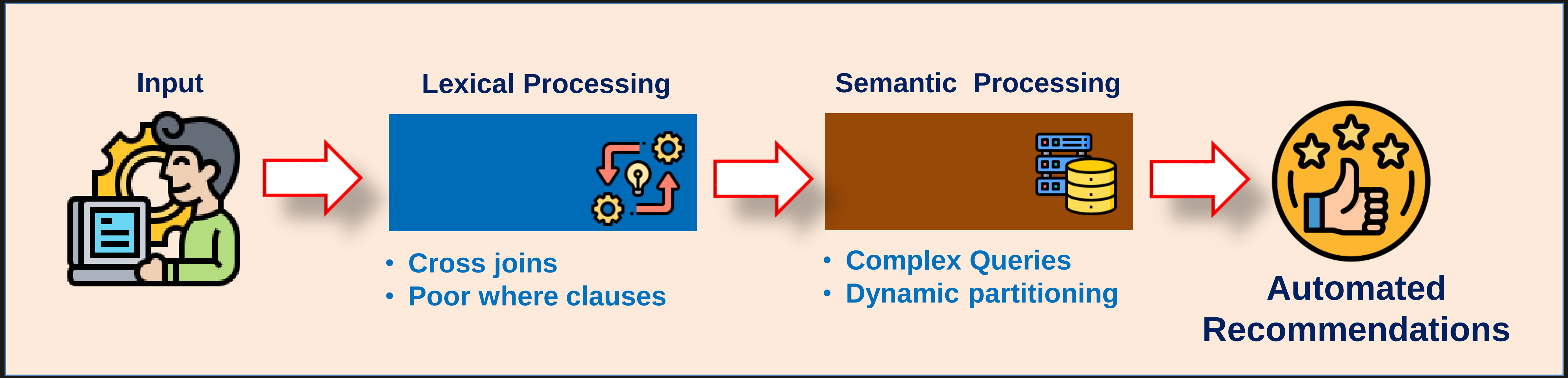
Govermation

SRE Approach to Dimension the Problem

Govermation

Big data query analyser developed by DBS

New Hadoop SQ Plug-in Architecture

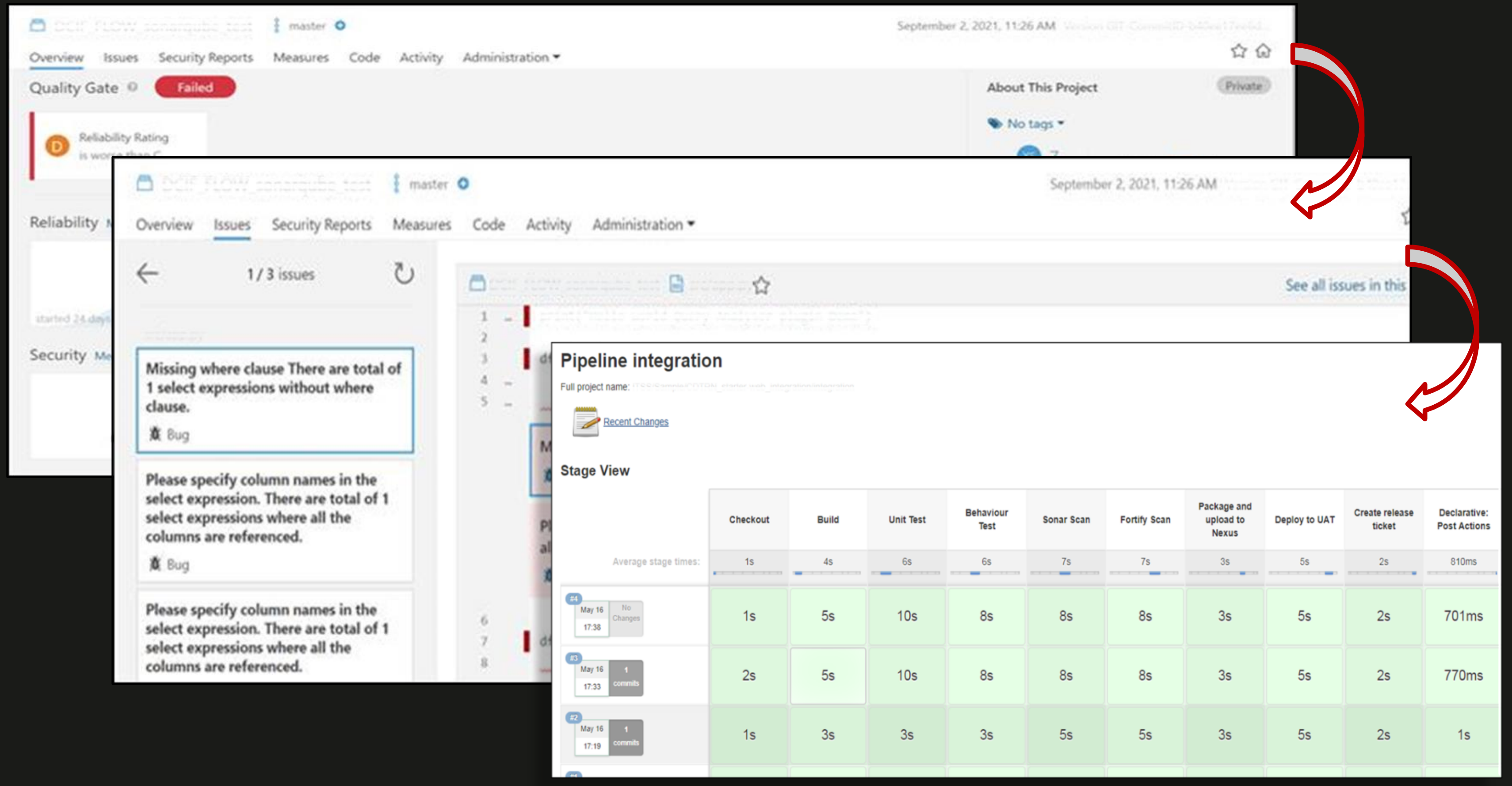


Lexical Processing:
- Cross joins
- Poor where clauses

Semantic Processing:
- Complex Queries
- Dynamic partitioning

Automated Recommendations

Query Analyser in Action



Missing where clause There are total of 1 select expressions without where clause.
Bug

Please specify column names in the select expression. There are total of 1 select expressions where all the columns are referenced.
Bug

Please specify column names in the select expression. There are total of 1 select expressions where all the columns are referenced.

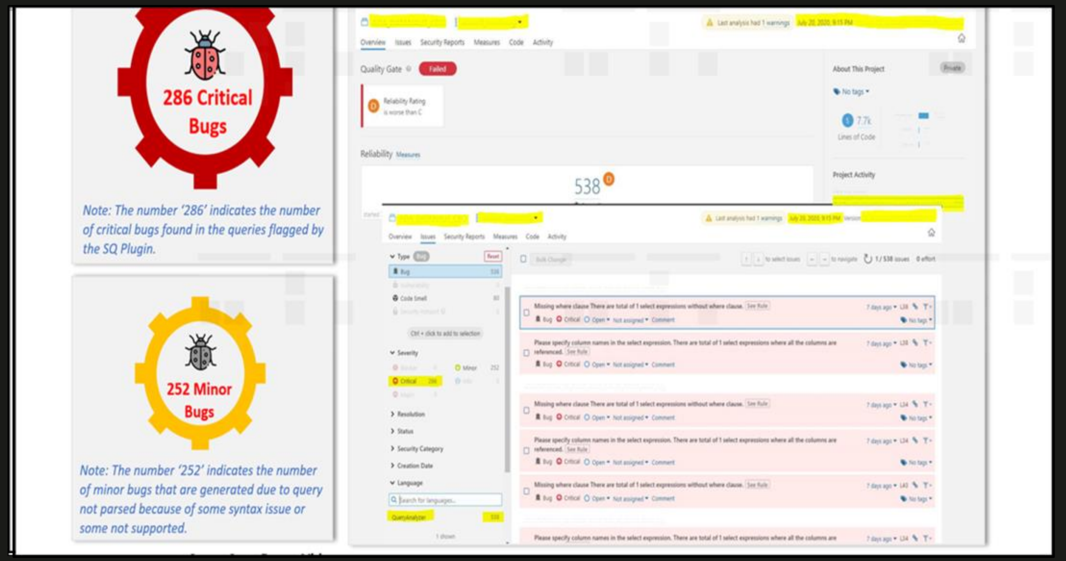
Pipeline integration

Full project name:

Recent Changes

Stage View

	Checkout	Build	Unit Test	Behaviour Test	Sonar Scan	Fortify Scan	Package and upload to Nexus	Deploy to UAT	Create release ticket	Declarative: Post Actions
Average stage times:	1s	4s	6s	6s	7s	7s	3s	5s	2s	810ms
#4 May 16 17:38 No Changes	1s	5s	10s	8s	8s	8s	3s	5s	2s	701ms
#3 May 16 17:33 1 commits	2s	5s	10s	8s	8s	8s	3s	5s	2s	770ms
#2 May 16 17:19 1 commits	1s	3s	3s	3s	5s	5s	3s	5s	2s	1s



Note: The number '286' indicates the number of critical bugs found in the queries flagged by the SQ Plugin.

Note: The number '252' indicates the number of minor bugs that are generated due to query not parsed because of some syntax issue or some not supported.

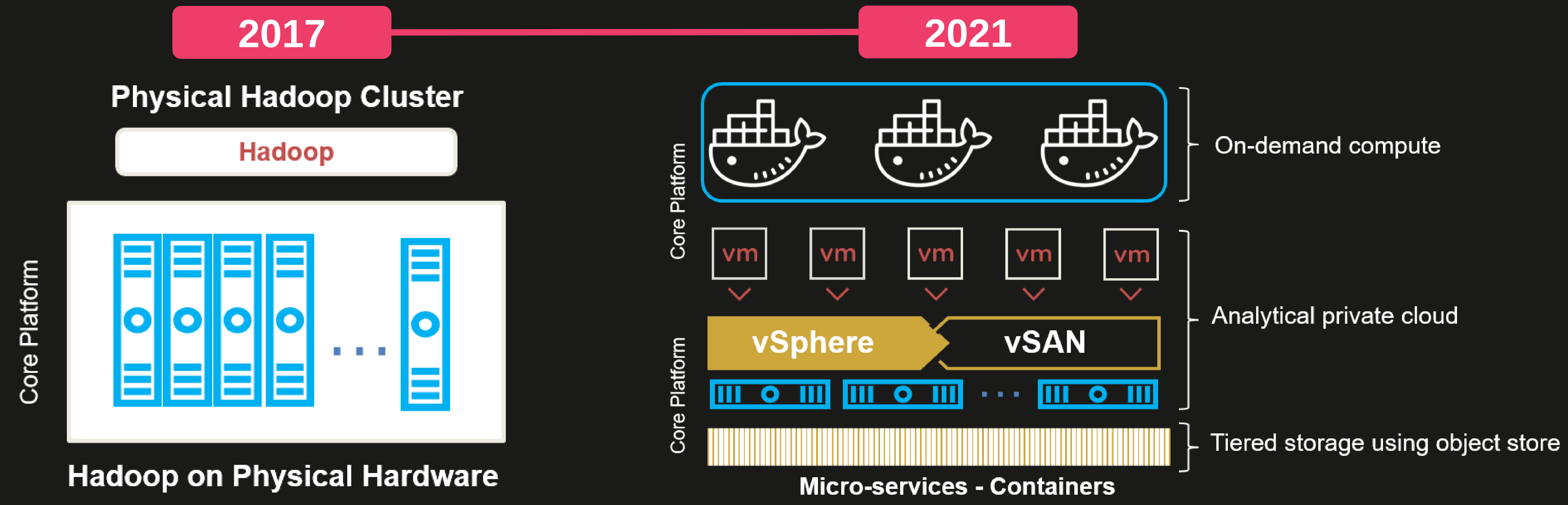
Missing where clause There are total of 1 select expressions without where clause.

Please specify column names in the select expression. There are total of 1 select expressions where all the columns are referenced.

SRE Approach to Dimension the Problem

Deep Engineering

Data Platform @ Scale



Objectives

Engineer resilient and dependable data platform

Cost Optimisation

- 96% Provisioning Improvement for persistent workload
- On-demand Compute in minutes

Productivity Improvement

- 60% batch run-time improvement

Agility

- 10x read and 2.17x write improvements

Risk Reduction

- 100% risk reduction

Joyful Customer Experience

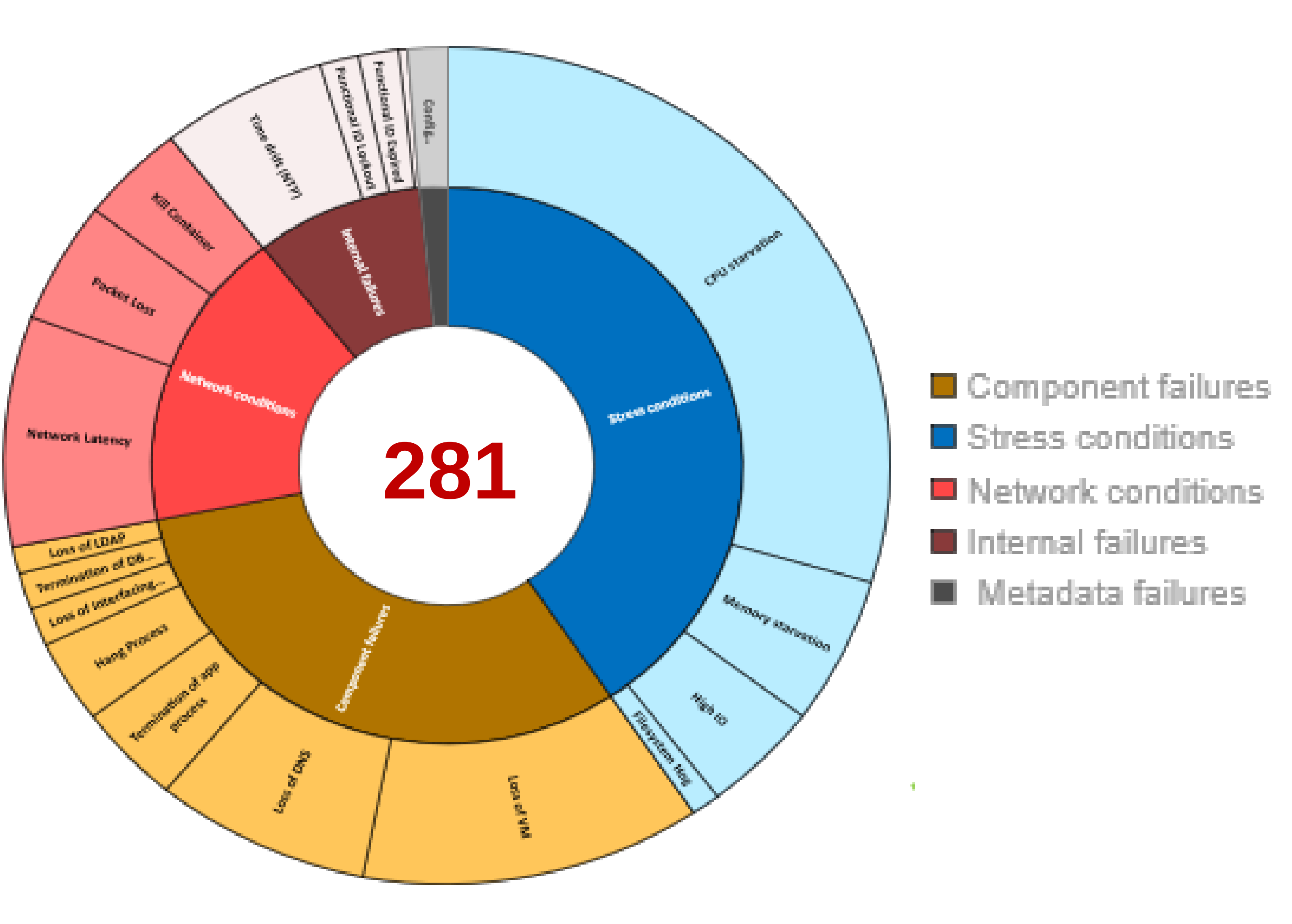
- Joyful experience

Chaos @ Scale

Our in-house chaos engineering tool, Wreckoon verified our deep engineering efforts could withstand different destructive and possible real-world scenarios.

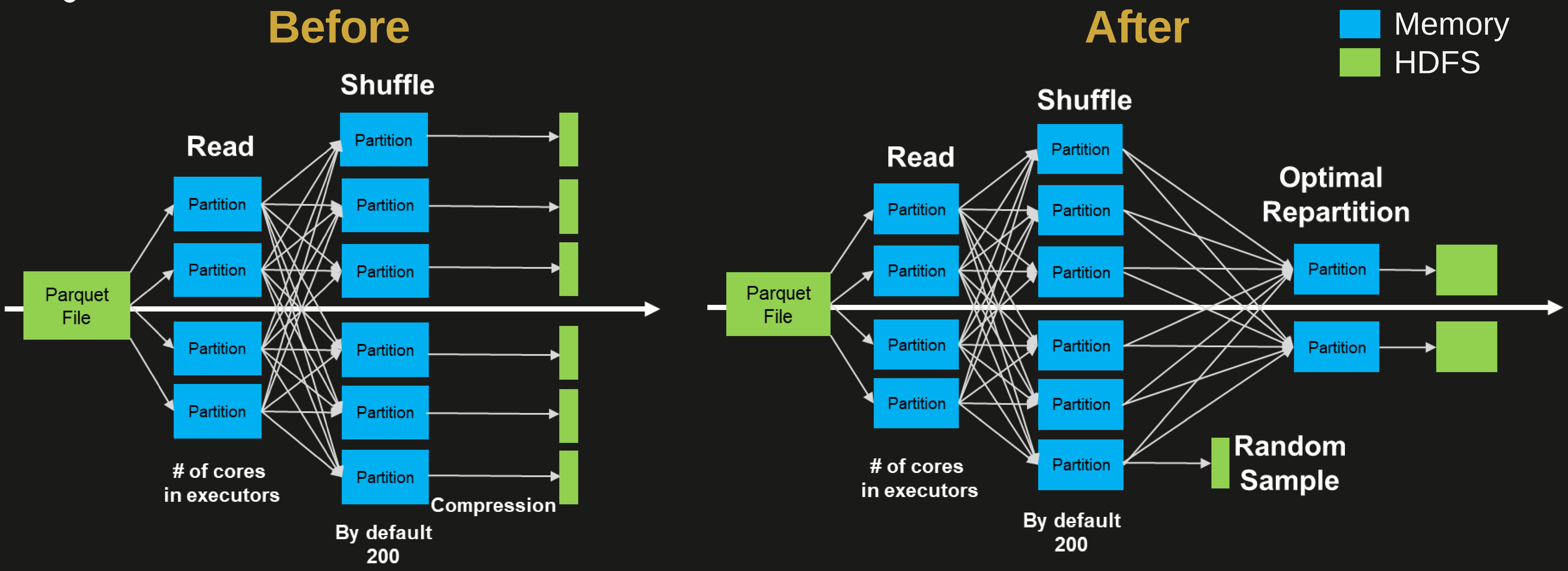
Attack Category	Attack details	Pass/Fail
Stress conditions	CPU starvation	✓
	Memory starvation	✓
	High IO	✓
	Filesystem Hog	✓
Component failures	Loss of VM	✓
	Termination of app process	✓
	Termination of DB listener	✓
	Loss of DNS	✓
	Hang Process	✓
	Loss of LDAP	✓
	Loss of Interfacing System	✓
Network conditions	Network Latency	✓
	Packet Loss	✓
	Kill Container	✓
Internal failures	Time drift (NTP)	✓
	Certificate expiry	✓
	Functional ID Expired	✓
	Functional ID Lockout	✓
Metadata failures	Config corruption -MariaDB	✓

Data Platforms



A total of 281 chaos tests have been conducted.

Optimal Spark Partitioning



- Directly writing tables back to HDFS after computation may result in many small files
- Parquet compression makes this worse

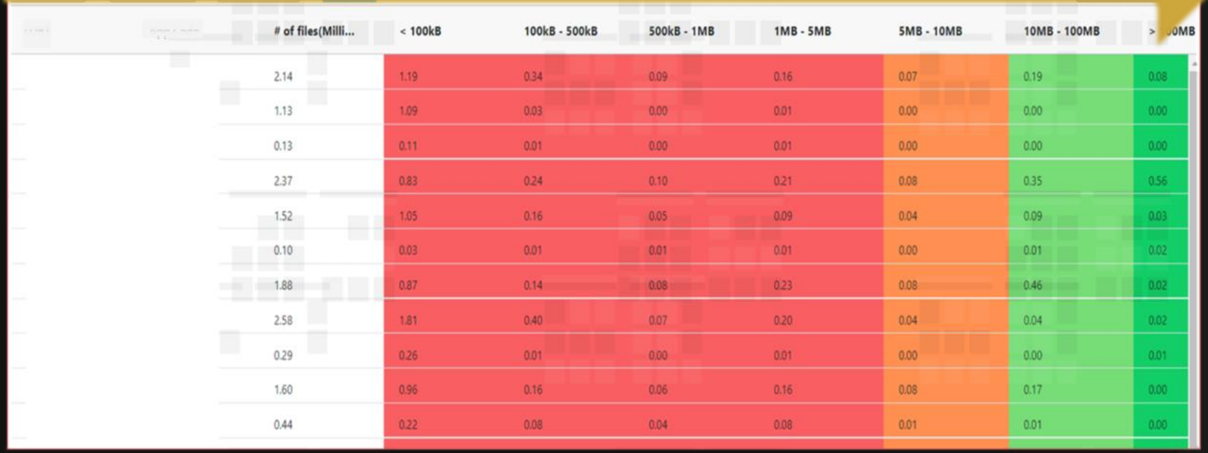
- Instead, we randomly write 10% of data to estimate final file size
- Then, we repartition to minimise small files before writing to HDFS

SRE Approach to Dimension the Problem

Collaboration

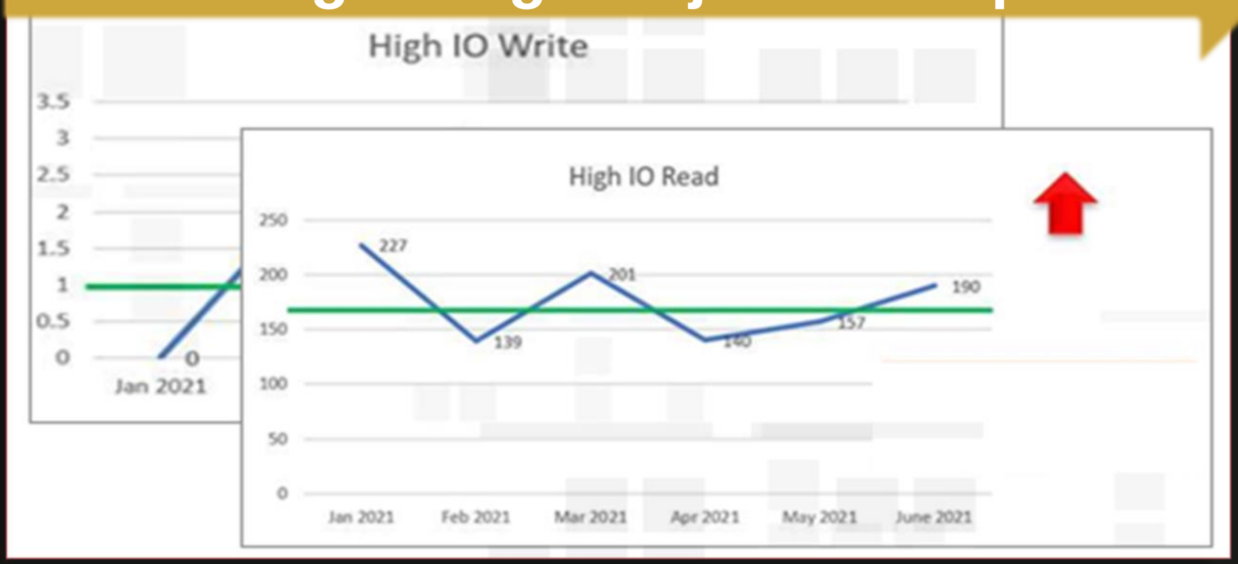
Building Ethos with Engineers to Track and Manage

Monitoring of small files & queries

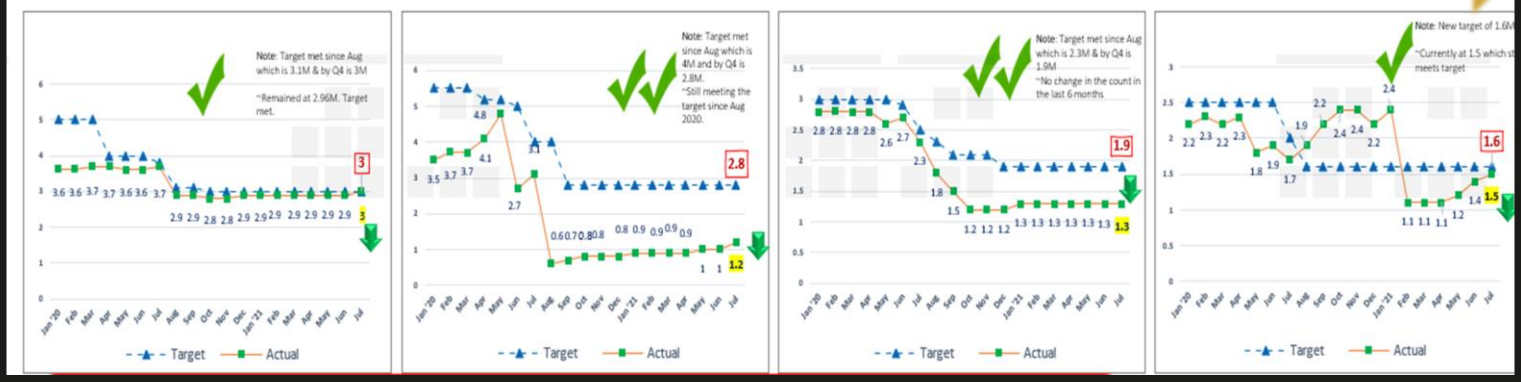


# of files(Milli...	< 100kB	100kB - 500kB	500kB - 1MB	1MB - 5MB	5MB - 10MB	10MB - 100MB	> 0MB
2.14	1.19	0.34	0.09	0.16	0.07	0.19	0.08
1.13	1.09	0.03	0.00	0.01	0.00	0.00	0.00
0.13	0.11	0.01	0.00	0.01	0.00	0.00	0.00
2.37	0.83	0.24	0.10	0.21	0.08	0.35	0.56
1.52	1.05	0.16	0.05	0.09	0.04	0.09	0.03
0.10	0.03	0.01	0.01	0.01	0.00	0.01	0.02
1.88	0.87	0.14	0.08	0.23	0.08	0.46	0.02
2.58	1.81	0.40	0.07	0.20	0.04	0.04	0.02
0.29	0.26	0.01	0.00	0.01	0.00	0.00	0.01
1.60	0.96	0.16	0.06	0.16	0.08	0.17	0.00
0.44	0.22	0.08	0.04	0.08	0.01	0.01	0.00

Monitoring of High IO jobs and queries



Burndown chart tracking



Note: Target met since Aug which is 3.1M & by Q4 is 3M

~Remained at 2.96M. Target met.

Note: Target met since Aug which is 4M and by Q4 is 2.8M.
~Still meeting the target since Aug 2020.

Note: Target met since Aug which is 2.3M & by Q4 is 1.9M
~No change in the count in the last 6 months

Note: New target of 1.6M

~Currently at 1.5 which st
meets target

SRE Approach to Dimension the Problem

Learning and Development

Continuous Learning and Certifications

SMALL FILES

AUTOMATED SCANS

QUERIES

SPARK

3 MAJOR TOPICS

DEVELOPER

5 DIGITAL LEARNINGS

BEST

700 TRAINED

CLASSROOM

DATABASE

HIVE

PRACTICES

PARTITIONS

HADOOP

HOUSEKEEPING

CODE ENHANCEMENT

Forward Looking

Long Term

How the new version of Hadoop (CDP) solves small file problems by implementing a new file system – Ozone (03FS)

Object Store optimised for big data, scales to billions of objects of all sizes. Key advantage of Ozone is that it segregates namespace and block space management.

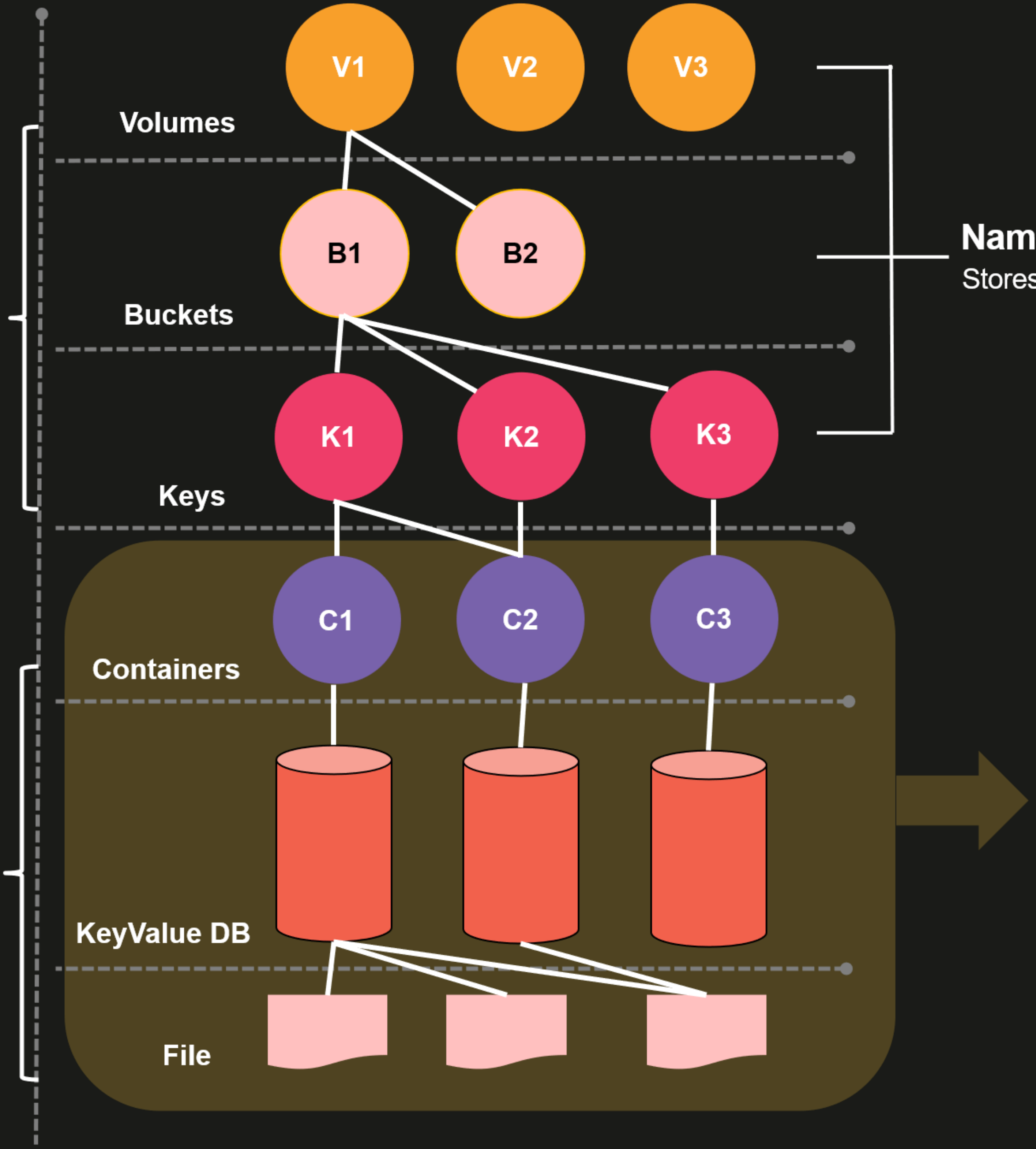
A. Ozone Manager (OM)
A namespace manager that implements the Ozone key-value namespace.

B. Hadoop Distributed Data Store (HDDS) – a generic distributed storage layer for blocks that does not provide a namespace.

Storage Containers - Key Value Store - It's a unit of replication

RAFT consensus protocol

Storage Container Manager (SCM) – A service that manages the lifecycle of containers.



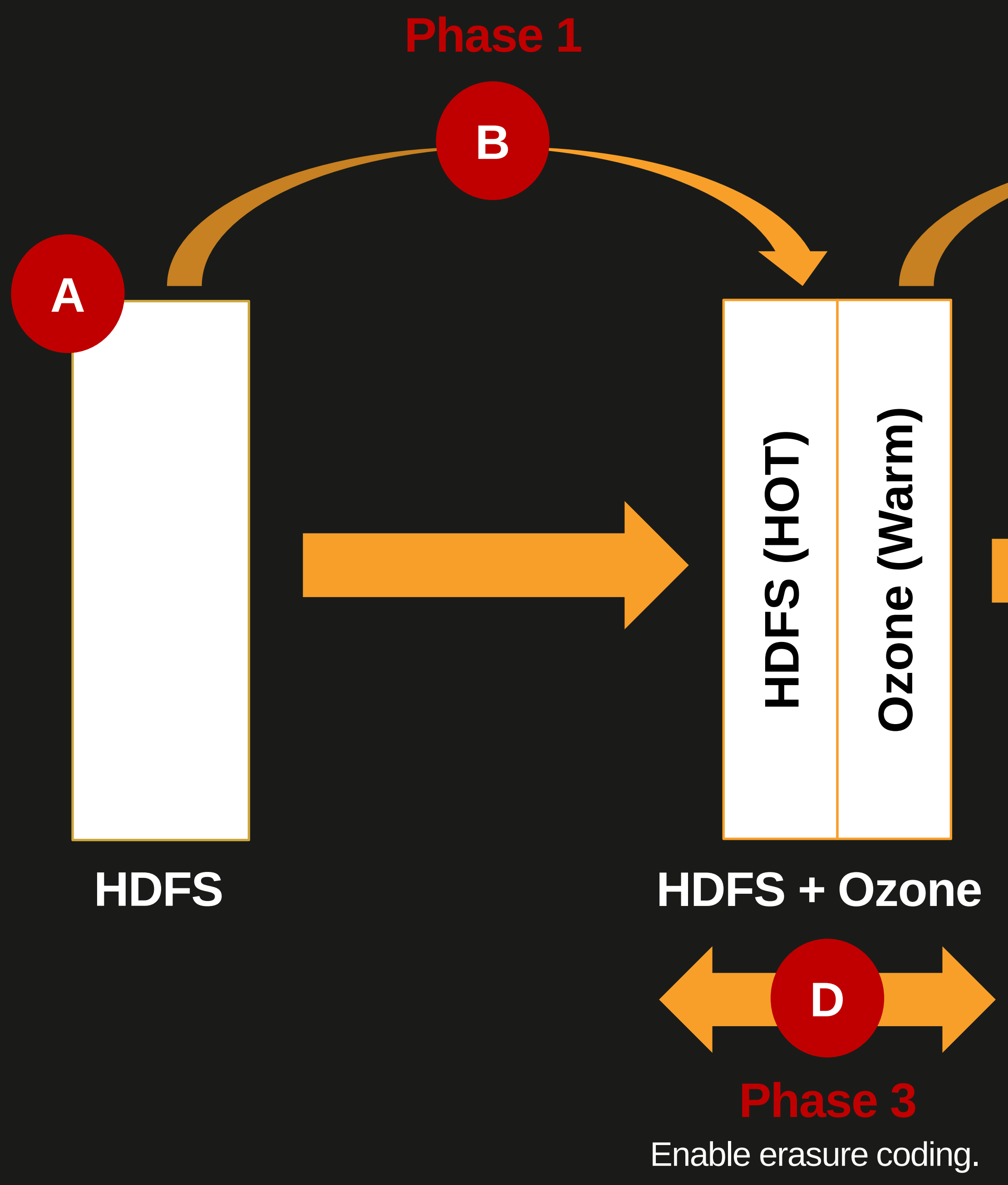
Name Space by Ozone Manager
Stores volumes, buckets and object metadata

Block space management by storage container manager

Pack multiple data blocks in a single storage unit i.e., when container is created, it starts in *open* state. When it is full (~ 5GB), the container closes.

Enabled Apps Can Leverage on Ozone and Seamless Archive in S3

Apps can leverage on policy such as temperature or business date to move data from hot to warm, and eventually to cold to manage the storage better.



Enable erasure coding.

A

- Current state with data stored in HDFS.

B

- Enable Ozone in separate data nodes.
- Policy-based migration to Ozone.
- This will be warm storage.

C

- Policy-based migration to S3 from Ozone.
- This will be cold storage.

D

- Enable erasure coding.

Key Takeaways

1. The presence of small files is not only one big boulder of a problem
2. There are some fundamental engineering concerns which we need to resolve in the SRE way
3. We need to implement technology interventions to reduce risks
4. The implementation of trainings for users increases awareness and adoption of best practices
5. Future tackling with experiments and adopting new solutions like Ozone