Of Mice and Elephants
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 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.
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**
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
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 = $((1024/128)*3+1)*150 = 3.7K \approx 0.0003\%$ overhead
- Files, $F$, of 1MB, heap requirement = $((1024/1)*3+1)*150 = 460K \approx 0.042\%$ overhead
- Files, $F$, of 1kb, heap requirement = $((1024/(1/1024)*3+1)*150 = 471MB \approx 45\%$ overhead

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

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

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

**Ingestion**
- Input files from sources are small
- Ingestion from Kafka
- Poor tuning practices during Ingestion
- No clear understanding of Bigdata tools
- Best Practices not followed

- ✓ Sqoop
- ✓ Spark
- ✓ Kafka
- ✓ Map-Reduce
- ✓ Nifi

**Transform**
- Application framework such as Spark
- Poor tuning practices for partitions during transformation
- No clear understanding of Bigdata tools
- Best Practices not followed

- ✓ Spark
- ✓ Map-Reduce

**Workflows**

**Small file contributors**

**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
- Education
- Design modern systems that are secure, scalable and built for failure
- Cultivate a culture and community with an SRE mindset
- Develop test effectiveness and coverage
- Increase test effectiveness and coverage
- Improve maintainability of code

Incident Response
- Respond and recover from failures and incidents efficiently
- Ability to launch software rapidly at scale through automation
- Build observability across multiple dimensions of infrastructure and applications

Best practices
- Blameless culture
- Engagement

Certification
- Sustainability
- Observability
- Collaboration

Governance
SRE Approach to Dimension the Problem

Governmation
Big data query analyser developed by DBS

New Hadoop SQ Plug-in Architecture

Query Analyser in Action
SRE Approach to Dimension the Problem

Deep Engineering
Objectives

Engineer resilient and dependable data platform

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

Productivity Improvement
- 60% batch runtime improvement

Agility
- 10x read and 2.17x write improvements

Risk Reduction
- 100% risk reduction

Joyful Customer Experience
- Joyful experience

Data Platform @ Scale

Physical Hadoop Cluster

Hadoop on Physical Hardware

2017

2021

On-demand compute
Analytical private cloud
Tiered storage using object store

Micro-services - Containers
**Chaos @ Scale**

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

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

A total of 281 chaos tests have been conducted.
Optimal Spark Partitioning

Before

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

After

- 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
- Monitoring of High IO jobs and queries
- Burndown chart tracking
SRE Approach to Dimension the Problem

Learning and Development
Continuous Learning and Certifications
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.
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.

- Current state with data stored in HDFS.
- Enable Ozone in separate data nodes.
- Policy-based migration to Ozone.
- This will be warm storage.
- Policy-based migration to S3 from Ozone.
- This will be cold storage.
- 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