Hadoop version 3.x

Agenda

- Overview of new features in Hadoop version 3
- Understanding the basics of erasure coding
- Understanding the features of intra data node balancer

Salient features Hadoop version 3

- **Stable version 3.0.0 available as on 13th December 2017**
- **Minimum required Java version is Java 8**
- **Support for erasure coding in HDFS**
- **YARN Timeline Service v.2**
- **Shell script rewrite**
- **Shaded client jars**
- **Intra-data node balancer**

A detailed description can be found the official apache Hadoop release page
https://hadoop.apache.org/docs/r3.0.0/

No need to save 3 replicas of the data

- Data was replicated 3 times in earlier version of Hadoop i.e. Hadoop version 1.x and 2.x
- The idea of replication this was to ensure data backup in the event of data node failure
- The drawback of this is that it reduces the storage space in the cluster by 1/3rd
- Data need not be replicated 3 times, instead a mechanism called "erasure coding" is used to re-construct the lost data block using parity bits

Hadoop version 3 uses “erasure coding”

- Lost data can be reconstructed using parity bits concept
- This relives a maximum of 50% extra storage space in the cluster

Understanding parity bits encoding

- Let's consider the following data in binary format 110010010101
- Divide the data into 3 parts (blocks) and name it as A, B and C

A : 1100	B : 1001	C : 0101

- Assume each block is of size 128MB
- 3 blocks stored as is would occupy 128*3 = 384MB
- On replicating it 3 times for failover recovery , it occupies = 1152 MB

Understanding XOR logic

A	B	Output
1	1	0
0	0	0
1	0	1
0	1	1

Calculating the parity block

A : 1100

B : 1001

C : 0101

Block A	Block B	Parity Block (using XOR logic)
1	1	0
1	0	1
0	0	0
0	1	1

Block A Is lost	Block B	Parity Block (using XOR logic)
X	1	0
X	0	1
X	0	0
X	1	1

The lost bits in block A can be reconstructed from the parity block

Reconstructing the lost data the parity block

A : 1100

Original contents of block A

A	B	Output
1	1	0
0	0	0
1	0	1
0	1	1

Recall XOR ….

Block A Is lost	Block B	Parity Block (using XOR logic)
X	1	0
X	0	1
X	0	0
X	1	1

Block B	Parity Block	**Applying XOR**
1	0	1
0	1	1
0	0	0
1	1	0

Similarly the block B data can also reconstructed if its lost as long as we have the other data block and parity intact

Saving the data & the parity blocks

A : 1100

B : 1001

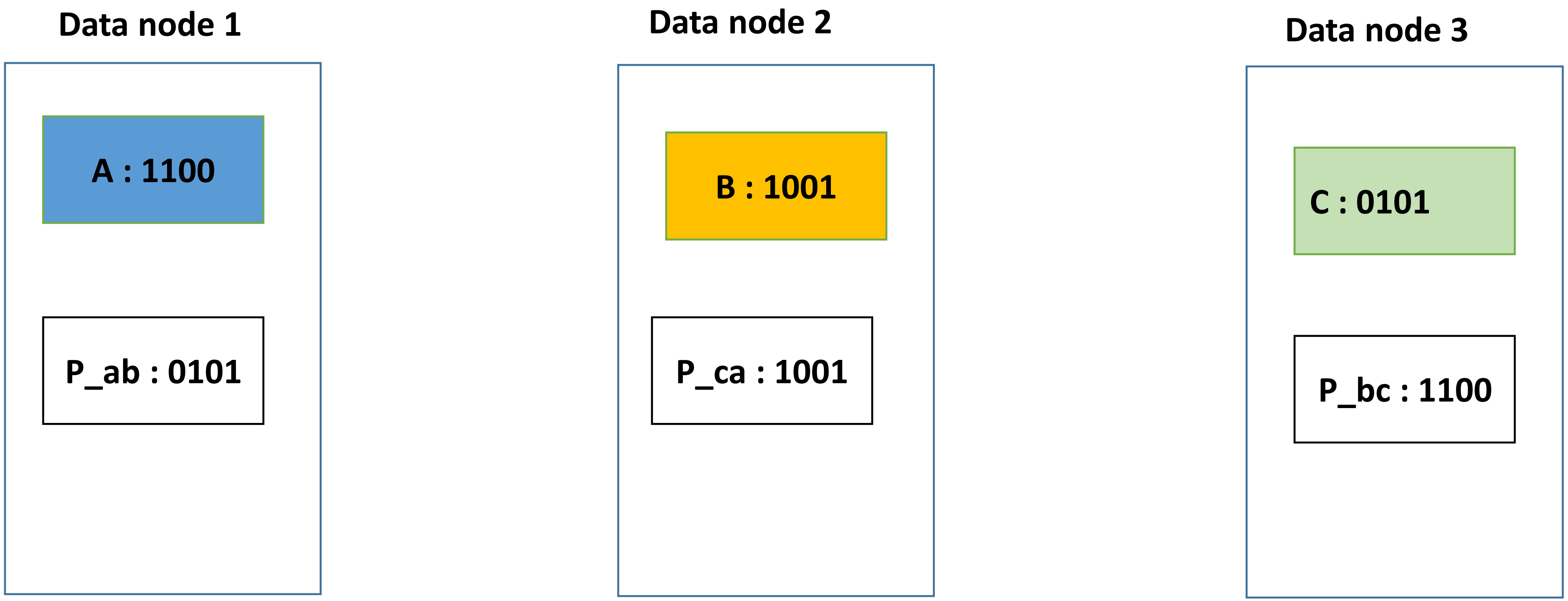
C : 0101

P_ab : 0101

P_bc : 1100

- We now have 5 blocks of information eventually (3 data blocks and 2 parity blocks)
- Each block occupies 128MB of data on the disc of a data node
- The 3 data blocks would occupy 128 * 3 = 384 and replicating it 3 times occupies 1152 MB
- Saving the above 5 blocks would occupy 128 * 5 = 640MB (almost 40% reduction in disc space)
- Parity blocks are in fact replicated directly or indirectly which would still give up to 30% space saving
- Any savings in disc space would directly reflect on the hardware cost savings

Saving the data blocks and the parity blocks



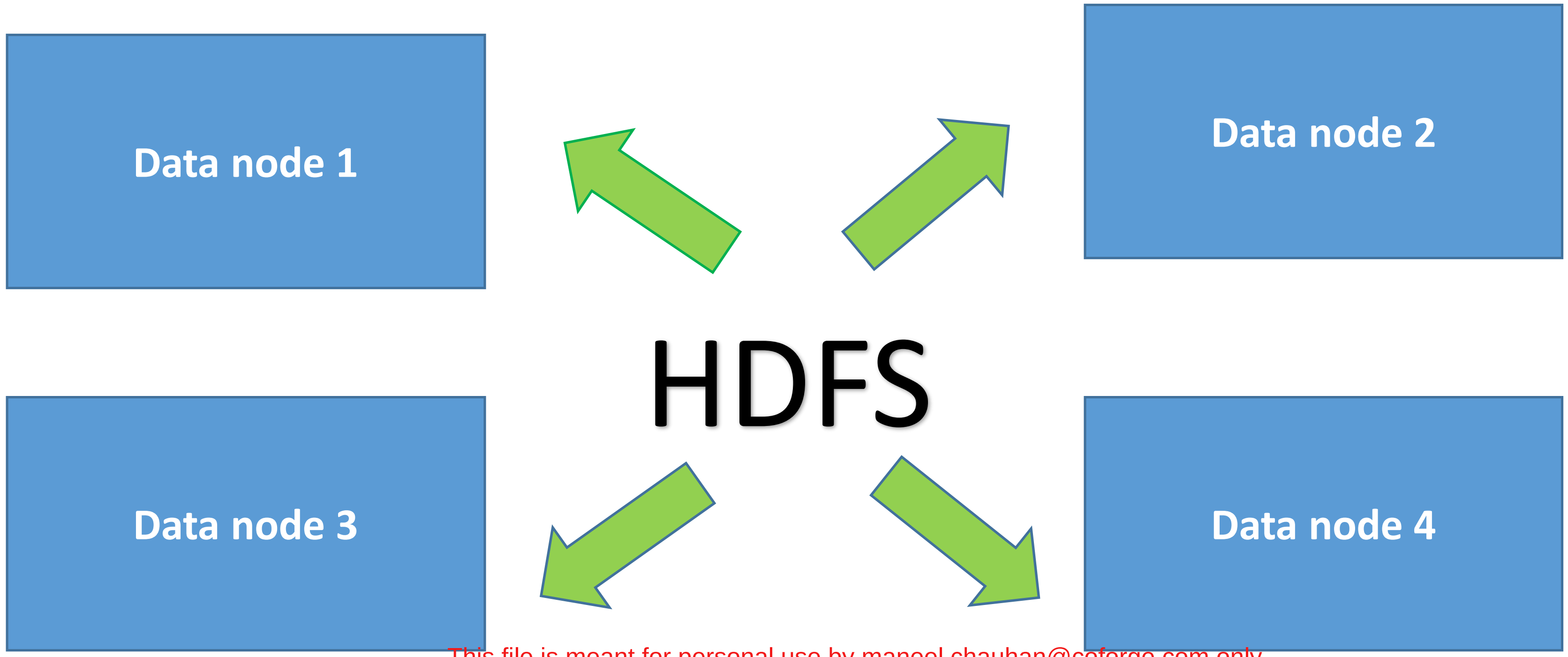
Note: Erasure coding uses a more complex technique to handle failure recovery using this parity bit concept. The discussion here is limited to basic understanding of block reconstruction.

Intra data node balancer

- **Earlier versions of Hadoop had an inter data node balancer (HDFS balancer)**
- **This ensured that the data in the cluster was always evenly distributed among the nodes**
- **In case we had 10GB of data and 5 data nodes, each data node would get 2GB of data**
- **Typically each data node in the Hadoop cluster has several hard discs**
- **Intra data node balancer ensures that data is evenly distributed among these nodes**

HDFS balancer

- **HDFS balancer ensures that the data is evenly distributed across the data nodes in a cluster**
- **An example, 12TB of data would be distributed as 3 TB on each of the four data nodes**
- **HDFS balancer would ensure deleting existing data or adding new data would still result in a balanced cluster**



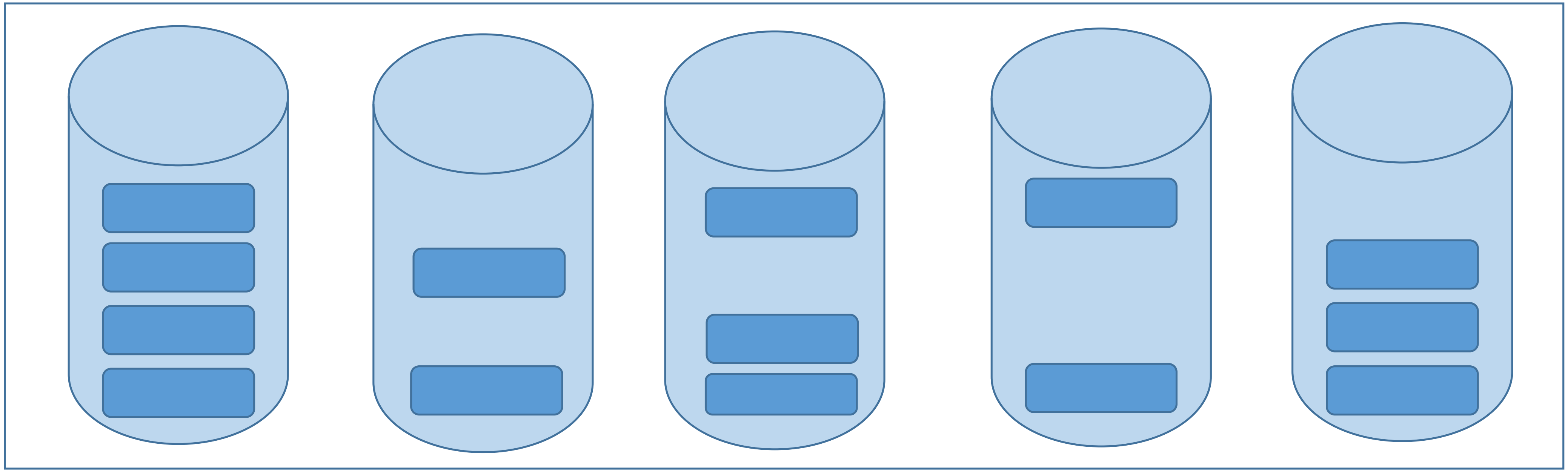
Drawbacks of HDFS balancer

- **A data node has multiple hard drives, usually in the order of 10 to 20 separate hard drives each of capacity anywhere between 1TB to 4TB**

- **HDFS balances does not ensure if the data in all the hard drives is evenly distributed**

- **The skewed data distribution in the hard drives of a data node is due to deletion of data and addition of new data into HDFS**

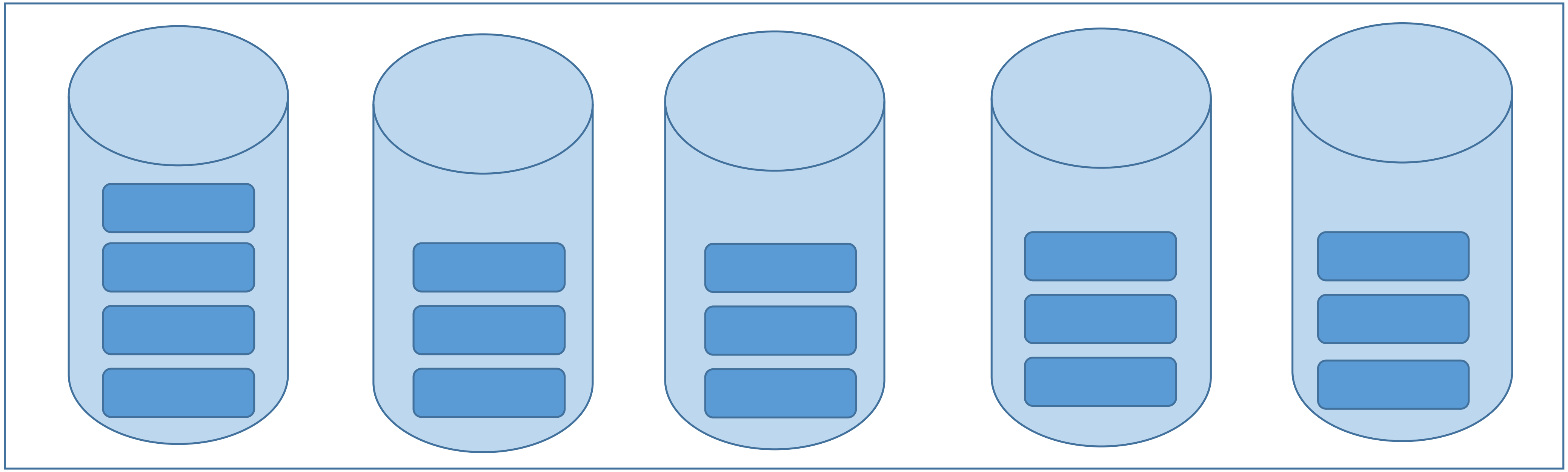
Data node with multiple hard drives



Intra data node balancer

- **This component ensures that the data is almost uniformly distributed among the all the discs of a data node**
- **This results in an improved data read performance**

Data node with multiple hard drives



Summary

- Understand the 2 important features of Hadoop version 3
- Basics of erasure coding
- Intra data node balancer