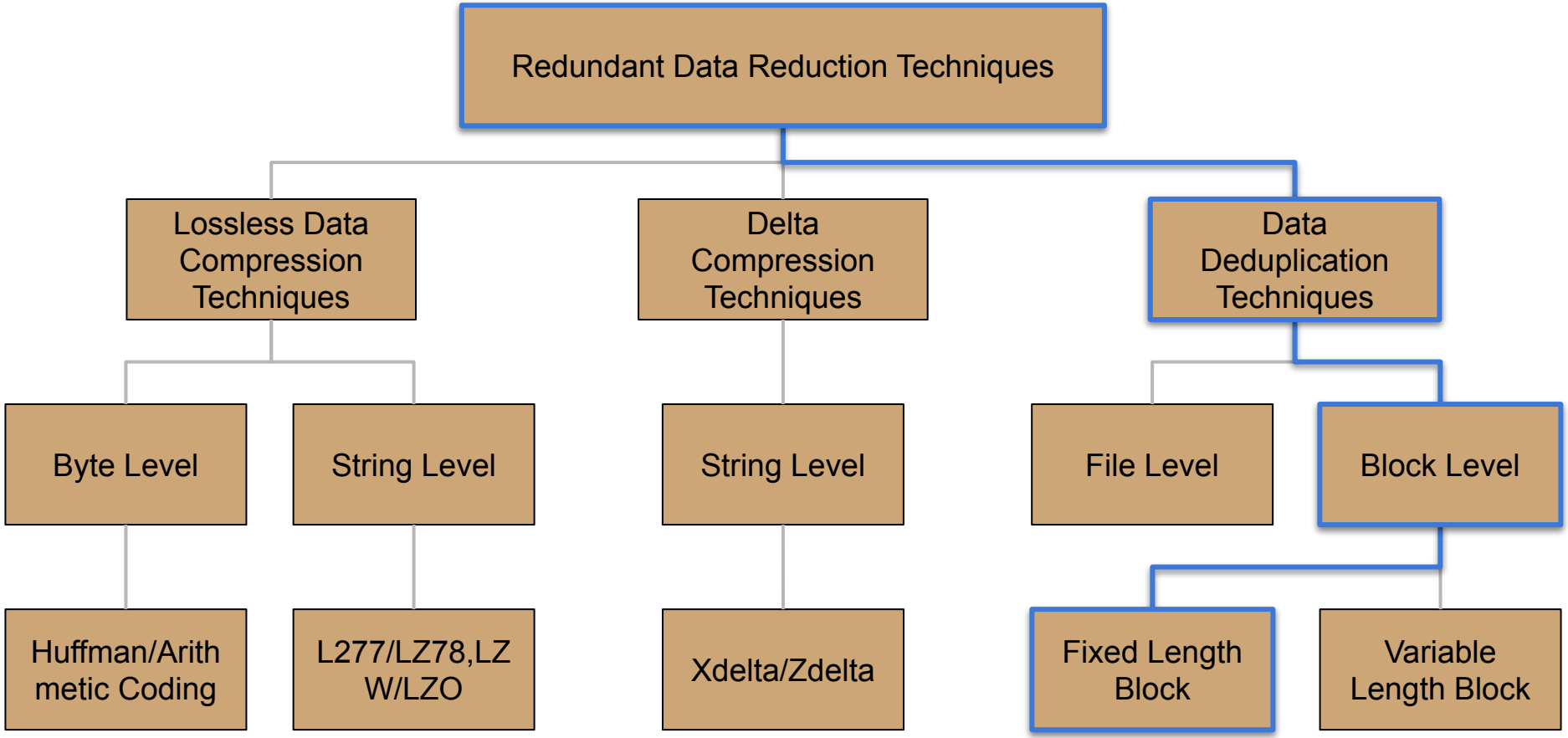
Data Deduplication on a Distributed File System

Keon, Ford, & Sam

Background

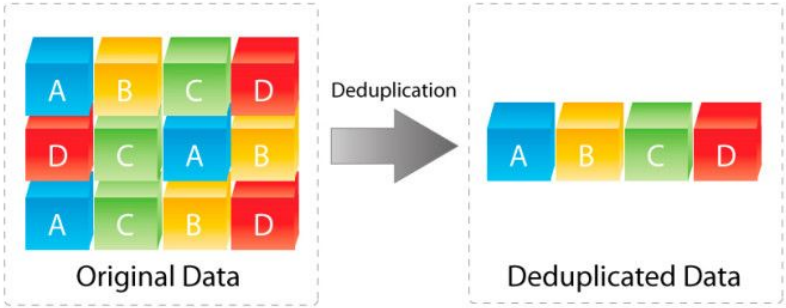
- Data is Growing:
  - People are generating 2.5 quintillion bytes of data each day
  - Nearly 90% of all data has been created in the last two years
- Real World Examples:
  - Up to 80% of some organizations' data is duplicated across the corporate network
  - Reducing deduplicated data can save money in terms of storage costs and backup speed
- Two Main Strategies: Data Compression vs. Data Deduplication
  - Data compression uses an algorithm to reduce the bits required to represent the stored data
  - Data deduplication: next slide


Redundant Data Reduction Techniques
Lossless Data Compression Techniques
Delta Compression Techniques
Data Deduplication Techniques
Byte Level
String Level
String Level
File Level
Block Level
Huffman/Arithmetic Coding
L277/LZ78,LZW/LZO
Xdelta/Zdelta
Fixed Length Block
Variable Length Block

Deduplication

What is data deduplication?

- Technique for eliminating redundant/duplicate data in a data set
- File Level vs. Block Level
- Fixed Length Block vs. Variable Length Block
- Inline vs. Post Processing
- Fingerprint Indexing vs. Other methods



Deduplication on Distributed File System

- Popular Distributed File Systems such as Hadoop tend not to have innate support for deduplication
  - Many files systems are build with other goals (fault tolerance, map reduce, etc.)
  - Implementing data deduplication on top of an existing file system is very difficult

- Attempts, however, have been made...
  - Droplet, Dedup, strategy using bloom filter etc.

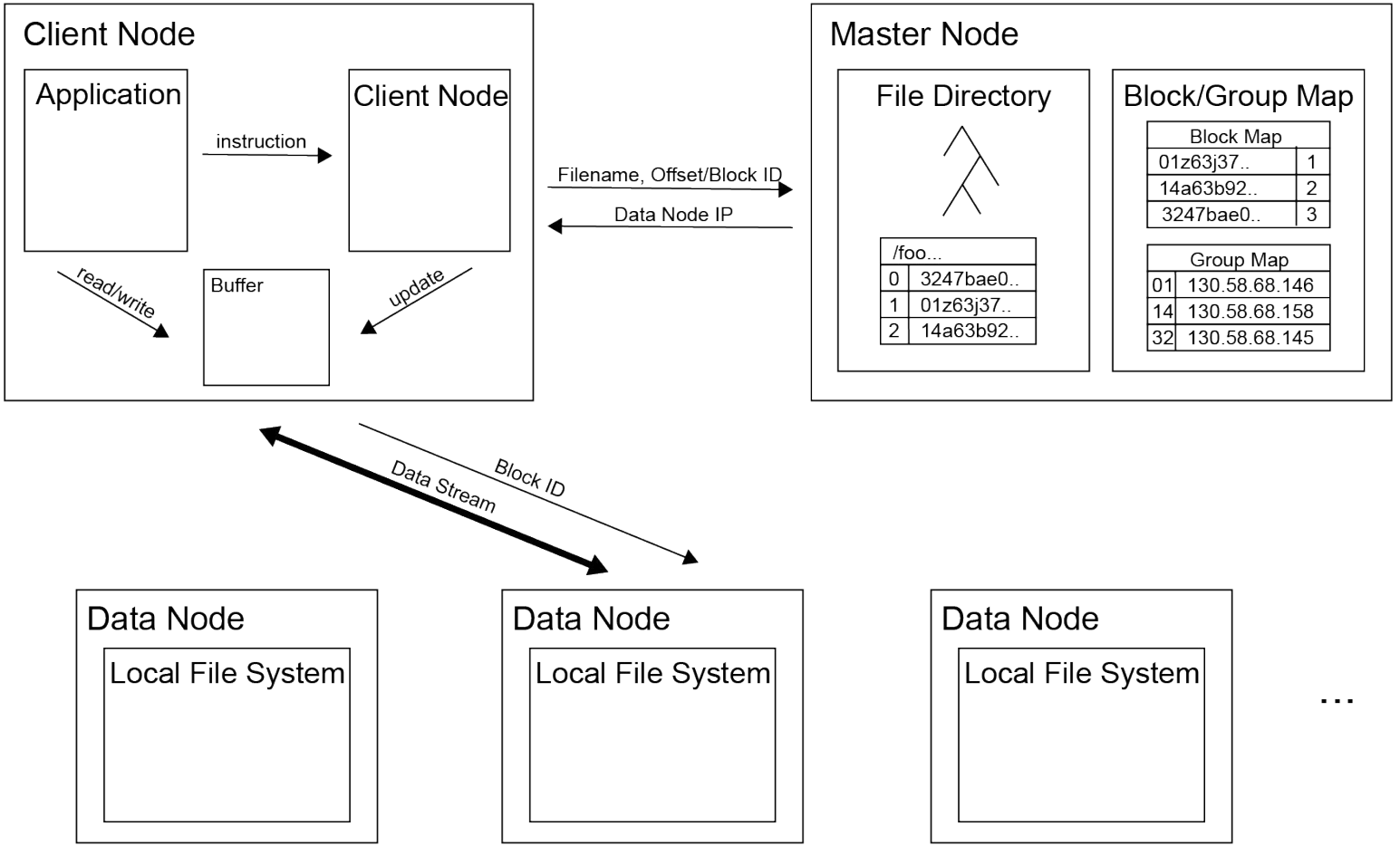
However..

- Two separate levels of interface and metadata
  - Deduplication is built on top of the file system
  - May compromise the performance of key operations (read, write, etc.)

- Hash collision → Highly unlikely, but what if?
  - MD5 and SHA-1 have already be deprecated
  - Is SHA-2 safe?

Our Strategy

- Similar architecture as HDFS and Google File System
  - Master Worker Model
  - Simple Coherency Model
- Build deduplication function into the distributed file system
- Each file is broken down to blocks
- Hash value of each block is computed using SHA-1 and used as the unique block ID/identifier
- Each block is stored in a data node and master node maintains the metadata
- If hash collision occurs, a byte-by-byte comparison is made


Client Node
Application
instruction
Client Node
read/write
Buffer
update
Filename, Offset/Block ID
Data Node IP
Master Node
File Directory
/foo...
0 3247bae0..
1 01z63j37..
2 14a63b92..
Block/Group Map
Block Map
01z63j37.. 1
14a63b92.. 2
3247bae0.. 3
Group Map
01 130.58.68.146
14 130.58.68.158
32 130.58.68.145
Block ID
Data Stream
Data Node
Local File System
Data Node
Local File System
Data Node
Local File System
...

Progress & //TODO:

- Progress
  - Implemented core file system functionality: open, close, read, write
  - Fully debugged and ready for testing!
- //TODO:
  - Start midway report
  - Design different kinds of experiments
  - Build scripts to run the experiments
  - Run the experiments
  - Write final report
  - Work on final presentation
  - Clean up source code
  - .....

Difficulties

- Debugging
- Creating hash functions
- Making Makefile
- C++
- Designing experiments
- Writing scripts to run them

Thank you!!