PACM: A Prediction-based Auto-adaptive Compression Model for HDFS

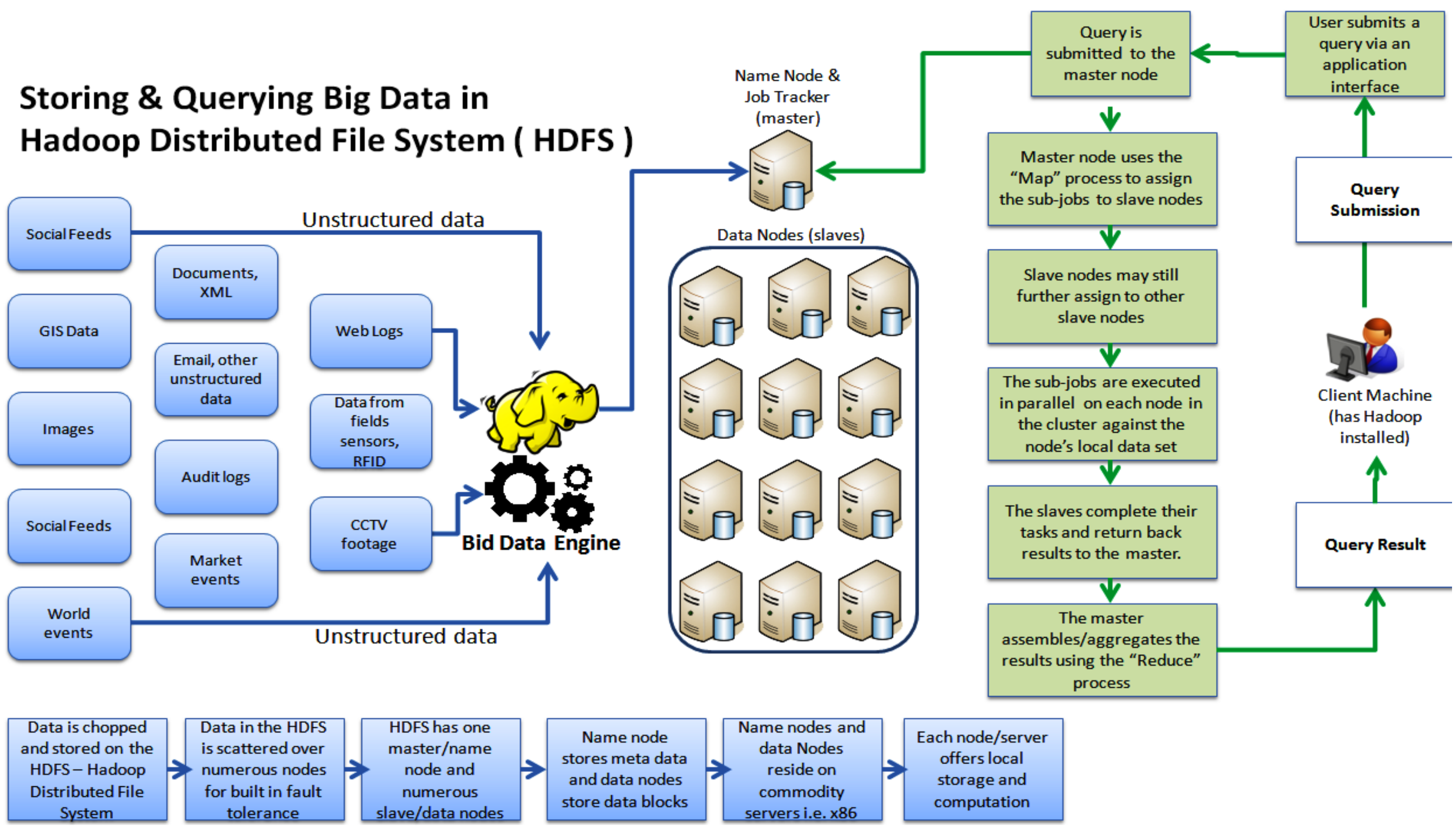
Ruijian Wang, Chao Wang, Li Zha

University of Chinese Academy of Sciences

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

Hadoop Distributed File System

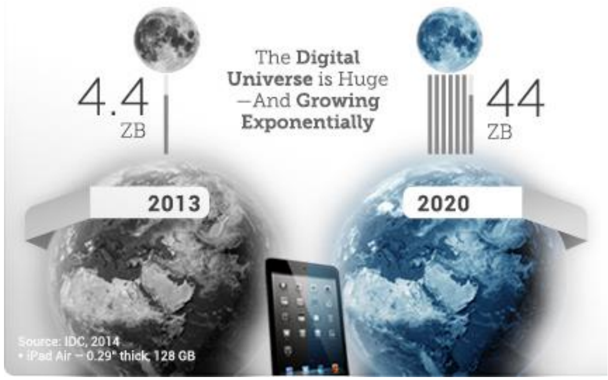
- Store a variety of data



http://popista.com/distributed-file-system/distributed-file-system:/125620

Mass Data

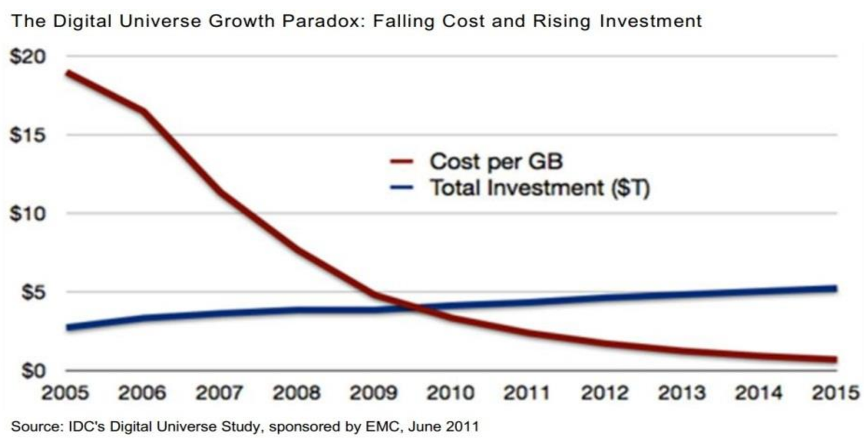
- The Digital Universe Is Huge –And Growing Exponentially[1]
- In 2013, it would have stretched two-thirds the way to the Moon.
- By 2020, there would be 6.6 stacks.



http://www.emc.com/collateral/analyst-reports/idc-digital-universe-2014.pdf

Motivation

- Compression can lead to improved I/O performance, and reduce storage cost.
- How to choose suitable compression algorithm in concurrent environment?



https://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf

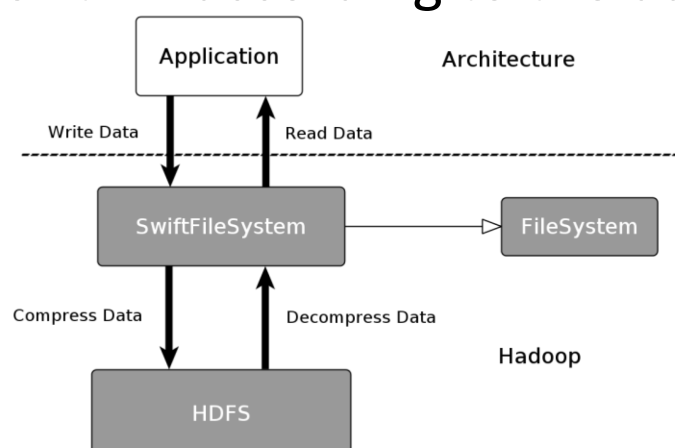
Related Work

- ACE [3] makes its decisions by predicting and comparing transfer performance for both uncompressed and compressed transfer.
- AdOC [4], [5] explores an algorithm that allows overlapping communication and compression and makes the network bandwidth fully utilized by changing the compression level.
- BlobSeer [2] By achieving compression on storage, reduce the space by 40%.

How can we use compression adaptively in HDFS to *improve the throughput* and *reduce the storage* while keeping the increasing weight *small*?

Solutions

- Build a layer between the HDFS client and the HDFS cluster to compress/decompress data stream automatically.
- The layer conducts compression by using an adaptive compression model : PACM.
  - Light weight : estimate parameters use sereval statistics
  - Adaptive: select algorithm according to the data and environment.



Results

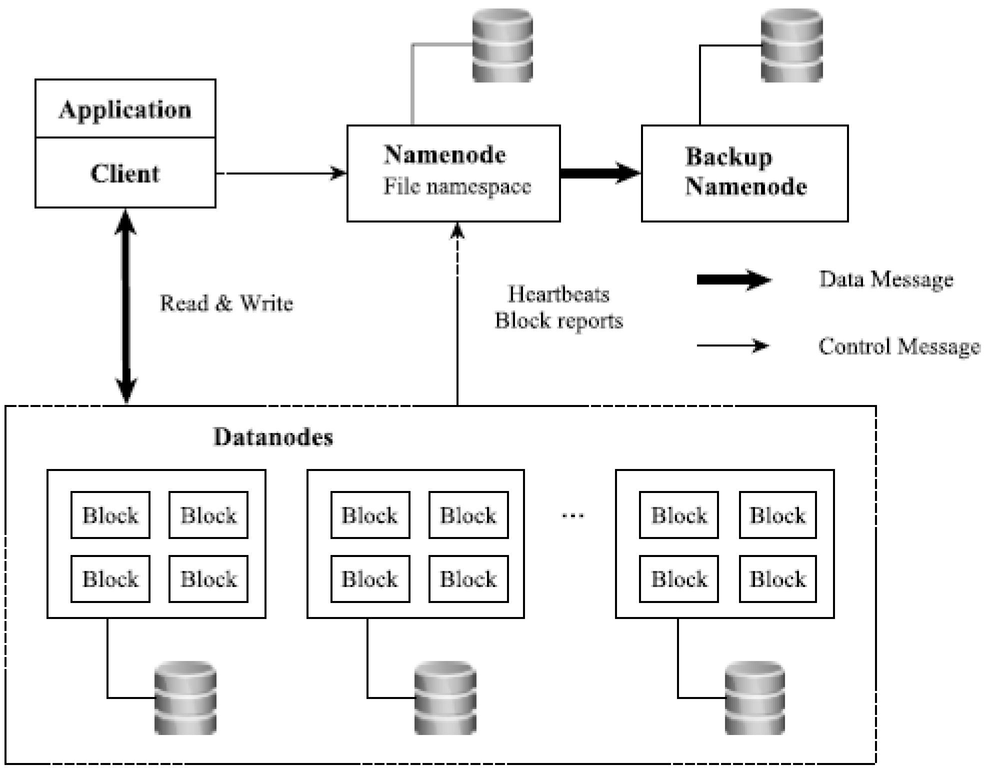
- The write throughput of HDFS has been improved by 2-5 times.
- Reduce the data by almost 50%.

Overview

- How HDFS work
- Challenges of compression in HDFS
- How to compress data: PACM
- Experiments
- Conclusion & Future work

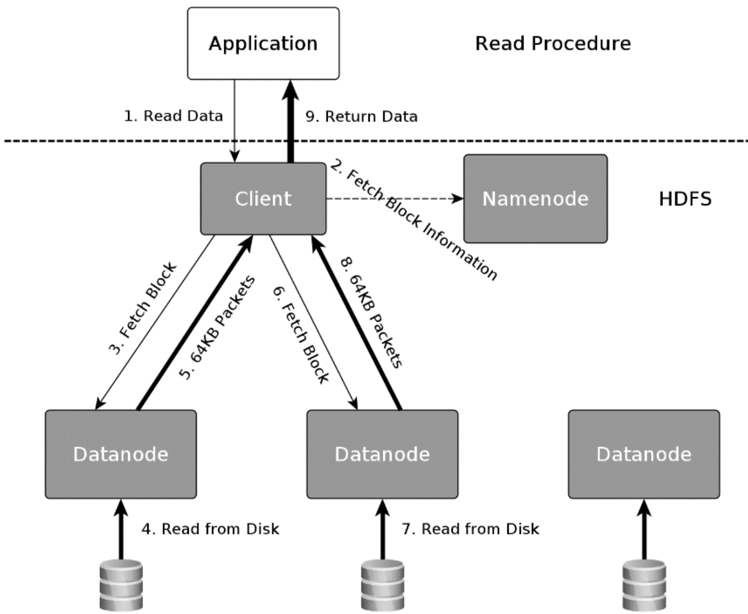
HDFS

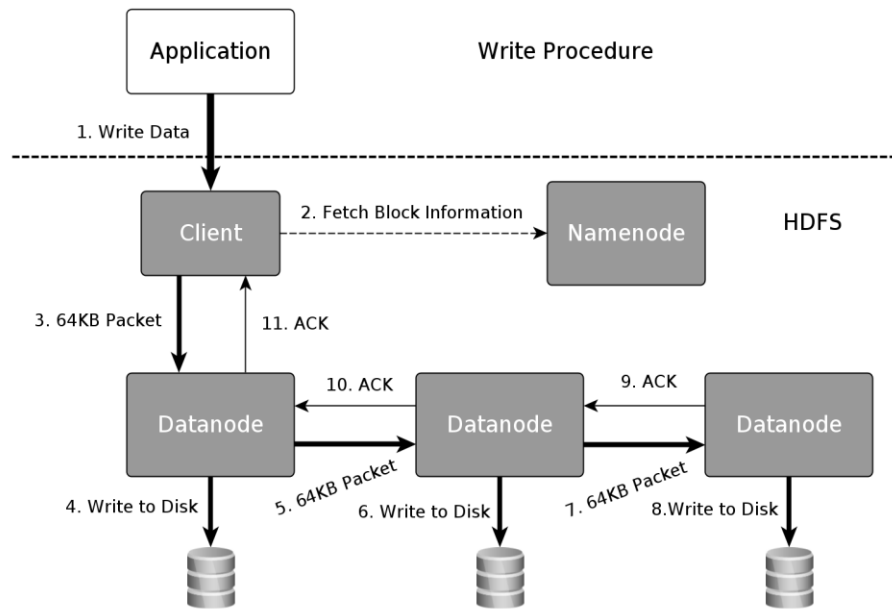
- Architecture
  - Consists of one master and many slave nodes



HDFS

- Read
- Write





Overview

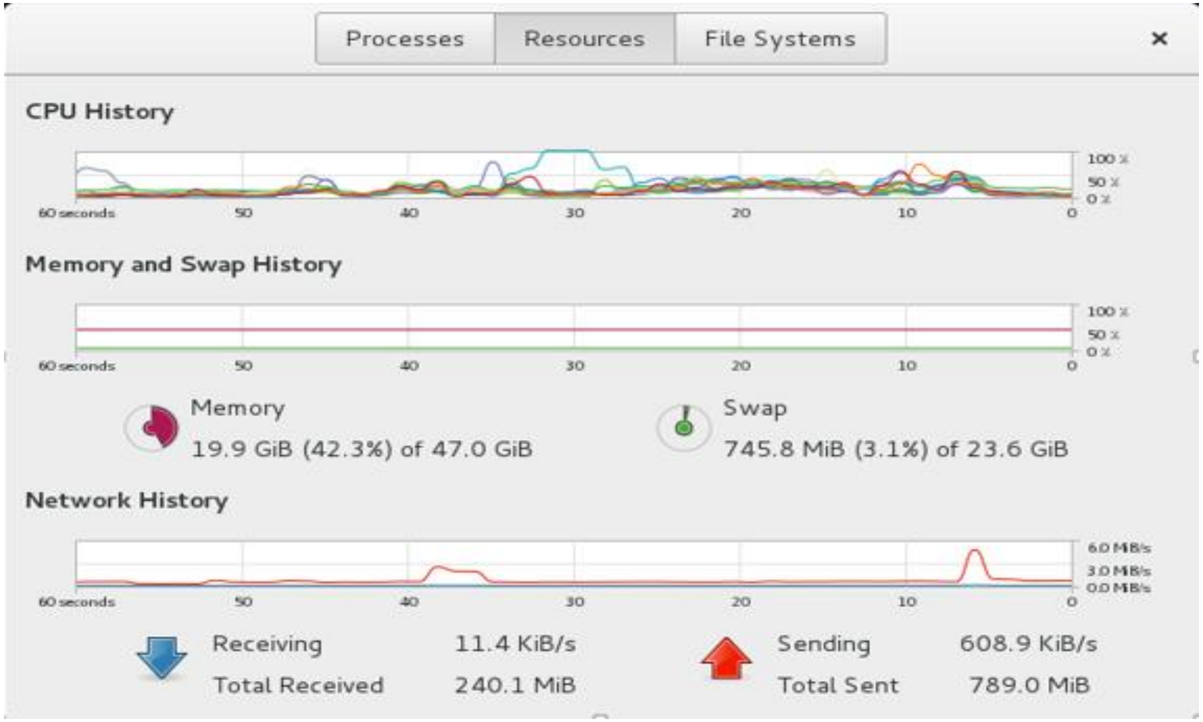
- How HDFS work
- Challenges of data compression in HDFS
- How to compress data: PACM
- Experiments
- Conclusion & Future work

Challenge#1

- Variable Data
  - Text
  - Picture
  - Audio
  - Video
  - …

Challenge#2

- Volatile Environment
  - CPU
  - Network Bandwidth
  - Memory
  - …

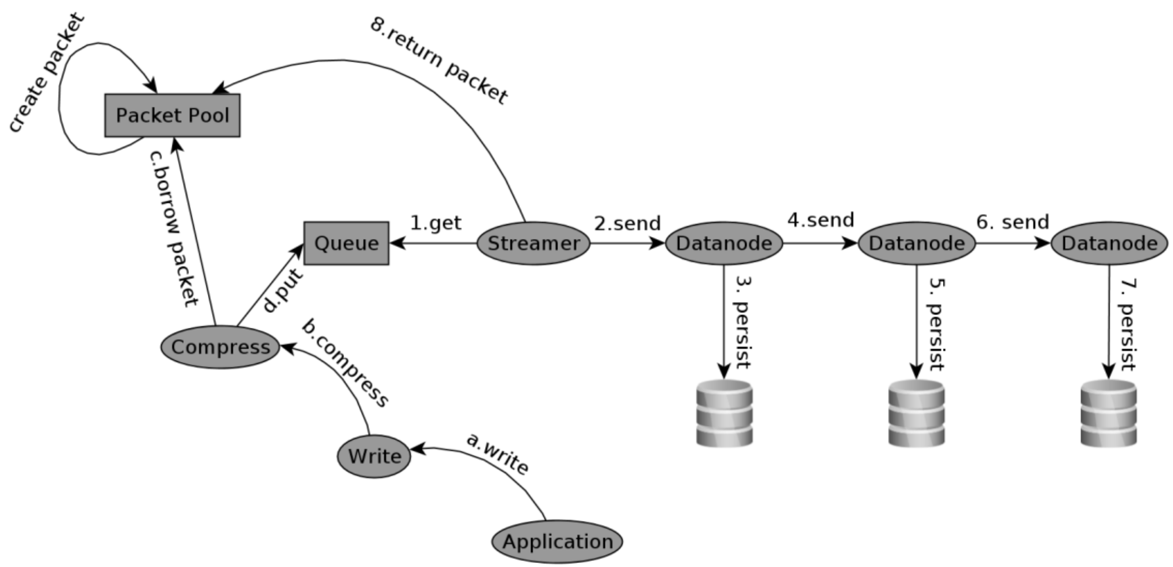


Overview

- How HDFS work
- Challenges of compression in HDFS
- How to compress data: PACM
  - *Compression Model*
  - *Estimation of compression ratio* $\boldsymbol{R}, \boldsymbol{CR}, \boldsymbol{TR}$
  - *Other evaluations*
- Experiments
- Conclusion & Future work

PACM: Prediction-based Auto-adaptive Compression Model

- Data processing procedure is regarded as a queue system.
- Introduce pipeline model into the procedure to speed up the data processing.



PACM: Prediction-based Auto-adaptive Compression Model

$$R = \frac{Compressed}{Uncompressed} \quad CR = \frac{Uncompressed}{CompressionTime} \quad TR = \frac{Data}{TransmissionTime}$$

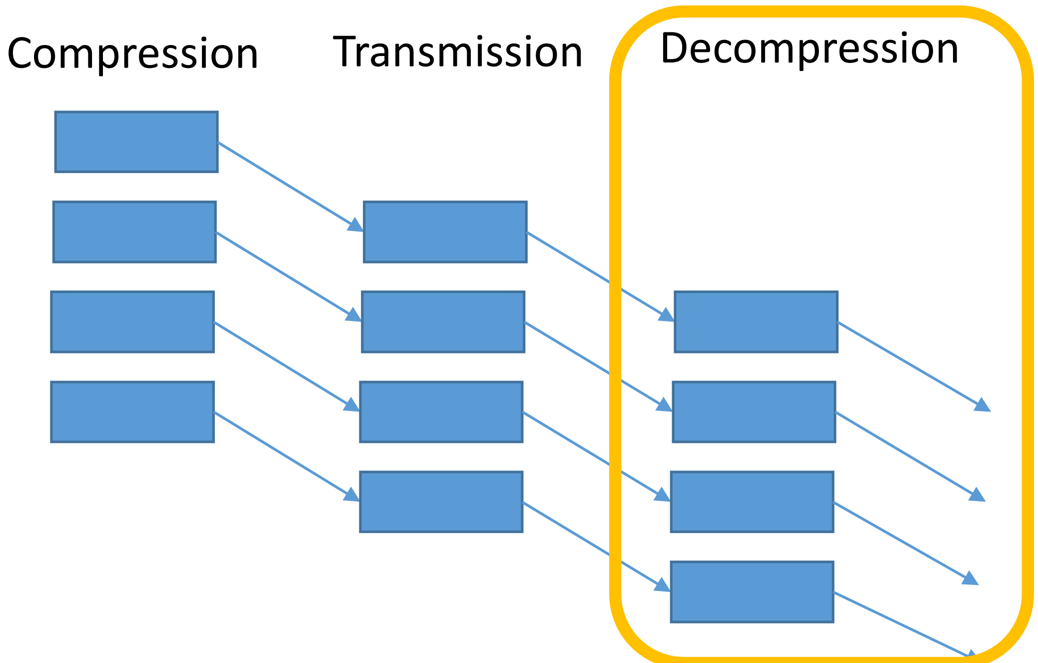
$$CT = \frac{B}{CR} \qquad DT = \frac{B}{DR} \qquad TT = \frac{B \times R}{TR}$$

Abbreviation	Elaboration
B	Block size
R	Compression ratio for a block
CR	Compression rate for a block
DR	Decompression rate for a block
CT	Compression time for a block
DT	Decompression time for a block
TR	Transmission rate
TT	Transmission time

PACM: Prediction-based Auto-adaptive Compression Model

- In pipeline model, T_p is the time a block spends in transferring from source to destination

$$T_p = \max\{CT, DT, TT\} = B \times \max\{\frac{1}{CR}, \frac{1}{DR}, \frac{R}{TR}\}$$



PACM: Prediction-based Auto-adaptive Compression Model

- [6] shows that HDFS I/O is usually dominated by Write operation due to the triplicated data blocks.

- ***Our model mainly focuses on HDFS write.***
- ***Presume that the decompression can be fast enough if the data is read.***

$$T_p = \max\{CT, TT\} = B \times \max\{\frac{1}{CR}, \frac{R}{TR}\}$$
$$\xRightarrow{minT_p} \frac{1}{CR} = \frac{R}{TR}$$

Key parameters

- compression ratio $\boldsymbol{R}$
- compression rate $\boldsymbol{CR}$
- transmission rate $\boldsymbol{TR}$

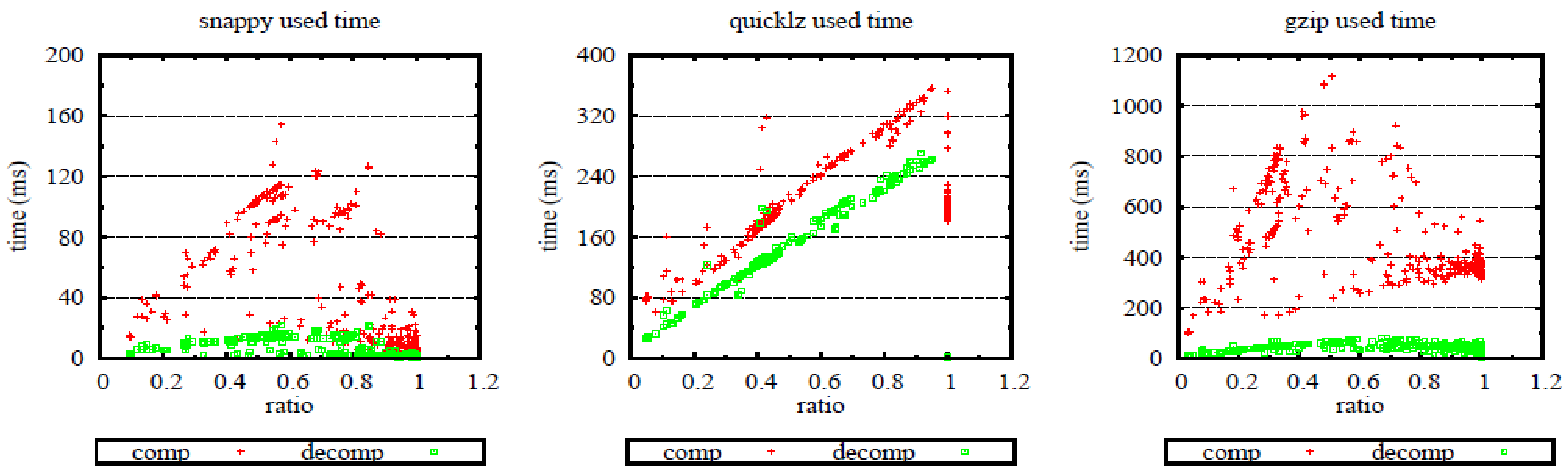
Estimation of compression ratio $\boldsymbol{R}$

- ACE makes a conclusion that there is an approximately linear relationship among the compression ratio of the different compression algorithms.

$$R_z = \frac{R_c - a_c}{b_c}$$

Estimation of Compression rate $\boldsymbol{CR}$

- We found that there is also an approximately linear relationship between the compression time and the compression ratio in each compression algorithm when the compression ratio is below 0.8.



Estimation of Compression rate $\boldsymbol{CR}$

- We defined the time of compressing 10MB data as CT
- $theoryCR_x$ may be quite different from the real value, which will increase the probability of wrong choice.
  - Introduced a variable $busy$ which refers to be busy degree of CPU.

$$CT_x = a_x + b_x \times R_x$$

$$theoryCR_x = 10 \times 1024 \times 1024 / CT_x$$

$$busy = \frac{lastCR_c}{theoryCR_c}$$

Estimation of Compression rate $\boldsymbol{CR}$

- Considering the deviation of calculation, we collected both the number of the blocks recently compressed(CNT) and the average compression rate($avgCR$) of each algorithm.

$$estCR_x = theoryCR_x \times busy \times \frac{100}{100 + CNT_x} + avgCR_x \times \frac{CNT_x}{100 + CNT_x}$$

Estimation of transmission rate TR

- According to the average transmission rate of recently transmitted 2048 blocks.

$$TR = \frac{\sum len_p}{\sum time_p}$$

Other Evaluations

- *Blocks of one batch (128 blocks)*
  - Use a batch as unit to avoid fluctuation of performance(for prediction is not precise).
- *Processing of original data*
  - Non-compression when R > 0.8 or CR < TR.
  - $UncompressTimes$ *(min 10, max 25)* record the number of batches written continuously by our model after entering into non-compression mode.

Summary of Estimation

- We make prediction based on the following formula and then update the algorithm before transmitting a batch of blocks to HDFS cluster.

$$T_p = \max\{CT, TT\} = B \times \max\{\frac{1}{CR}, \frac{R}{TR}\}$$
$$\Rightarrow \left|\frac{1}{\mathrm{CR}} - \frac{R}{TR}\right|$$

$$\xrightarrow{minT_p} |CR \times R - TR|, CR > TR\ and\ R < 0.8$$

Overview

- How HDFS work
- Challenges of compression in HDFS
- How to compress data: PACM
- Experiments
- Conclusion & Future work

Experimental Environment

EXPERIMENT ENVIRONMENT	
CPU	Intel(R) Xeon(R) CPU E5-2650 @ 2.0GHz * 2
Memory	64GB
Disk	SATA 2TB
Network	Gigabit Ethernet
Operating System	CentOS 6.3 x86_64
Java Run Time	Oracle JRE 1.6.0_24
Hadoop Version	hadoop -0.20.2-cdh3u4
Test File	1GB log +1GB random file +1GB compressed file
Hadoop Cluster A	
DatanodeNum	3
Disk	1
NIC	1
Hadoop Cluster B	
DatanodeNum	3
Disk	6
NIC	4

Experimental Environment

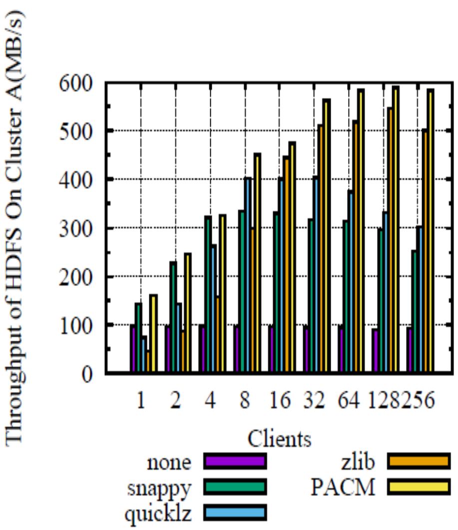
EXPERIMENT ENVIRONMENT(4 AWS EC2)	
CPU	Intel(R) Xeon(R) CPU E5-2680 @ 2.8GHz * 2
Memory	15GB
Disk	SSD 50GB
Network	Gigabit Ethernet
Operating System	Ubuntu Server 14.04 LTS
Java Run Time	Oracle JRE 1.7.0_75
Hadoop Version	hadoop -2.5.0-cdh5.3.0
Test File	24 * 1GB random file
Hadoop Cluster C	
DatanodeNum	3
Disk	1

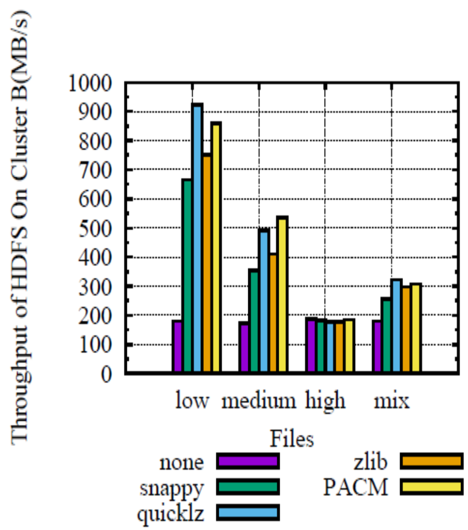
Workload

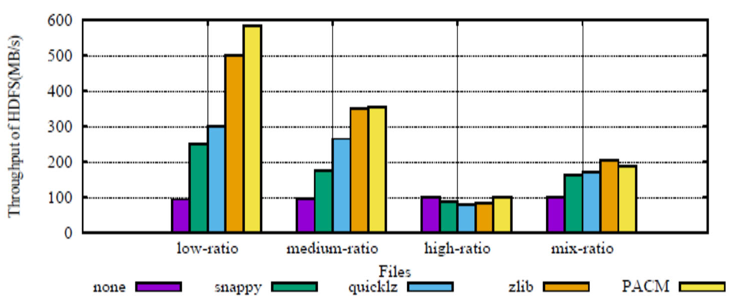
- HDFSTester
  - Different clients write
  - Write different files
- HiBench
  - TestDFSIOEnh
- RandomTextWriter
- Sort

Results

- Adapting to Data and Environment Variation
  - Variable clients on Cluster A
  - Variable compression ratio file on Cluster B
  - On average, PACM outperformed zlib by 21%, quicklz by 27% and snappy by 47%.







Results

- Validation for Transparency
  - The R of zlib, quicklz and snappy are 0.37, 0.51 and 0.61
  - HiBench
    - TestDFSIOEnh on Cluster B

Test / Algorithm	A(write)	B(read)
None	124.33	357.62
Zlib	175.26	1669.18
Quicklz	267.79	909.69
Snappy	222.41	2242.13
PACM	260.56	962.97

Results

- Validation for Transparency
  - RandomTextWriter
  - Sort
    - Sort A: all data is not compressed
    - Sort B: only input and output data is compressed
    - Sort C: only shuffle data is compressed
    - Sort D: input, shuffle and output data is compressed

job	None	Zlib	Quicklz	Snappy	PACM
RTW	221	140	105	131	107
Sort A	700	X	X	X	X
Sort B	X	515	433	419	427
Sort C	X	514	452	457	527
Sort D	X	366	294	312	411

Overview

- How HDFS work
- Challenges of compression in HDFS
- How to compress data: PACM
- Experiments
- Conclusion & Future work

Conclusion

- PACM shows a promising adaptability to the varying data and environment.
- The transparency of PACM could benefit the applications of HDFS.

Future work

- Have a combination model for both read and write.
- Design a model with low compression ratio and high throughput.
- Design a auto-adaptive compression model for MapReduce.

References

1. IDC, “The digital universe of opportunities: Rich data and the increasing value of the internet of things.” [Online]. Available:http://www.emc.com/collateral/analyst-reports/idc-digital-universe-2014.pdf

2. B. Nicolae, “High throughput data-compression for cloud storage,” in *Proceedings of the Third international conference on Data management in grid and peerto-peer systems*, ser. Globe’10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 112.

3. C. Krintz and S. Sucu, “Adaptive on-the-fly compression,” *Parallel and Distributed Systems, IEEE Transactionson*, vol. 17, no. 1, pp. 15–24, 2006.

4. E. Jeannot and B. Knutsson, “Adaptive online data compression,” in High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on, 2002, pp. 379–388.

5. “AdOC library ver. 2.2.” [Online]. Available: http://www.labri.fr/perso/ejeannot/adoc/adoc.html

6. T. Harter, D. Borthakur, S. Dong, A. Aiyer, L. Tang, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, “Analysis of hdfs under hbase: A facebook messages case study,” in Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14). Santa Clara, CA: USENIX, 2014, pp. 199–212.

Q&A

Thank you！