

# Optimistic Concurrency Control in a Distributed NameNode Architecture for Hadoop Distributed File System

Qi Qi

Instituto Superior Técnico - IST (Portugal)  
Royal Institute of Technology - KTH (Sweden)  
Swedish Institute of Computer Science - SICS (Sweden)

*qiq@kth.se*

19 September 2014

# Motivation

## Industrial Standard in Big Data Era

Apache Hadoop Ecosystem

## Limits to growth in HDFS

Number of Files	Memory Requirement	Physical Storage
1 million	0.6 GB	0.6 PB
<b>100 million</b>	<b>60 GB</b>	<b>60 PB</b>
1 billion	600 GB	600 PB

## Hops-HDFS and Its Limitation

Distributed NameNode Architecture

Maintain HDFS Strong Consistency Semantics

Concurrency Restricted

# Problem Statement

## HDFS

System-level Lock

## Hops-HDFS v1

System-level Lock

## Hops-HDFS v2

Row-level Lock

## MySQL Cluster

Read Committed / Anomalies



# Contribution

## Architectures and Namespace Concurrency Control

GFS, HDFS, Hops-HDFS and MySQL Cluster

## Performance Assessment and Limitation Analysis

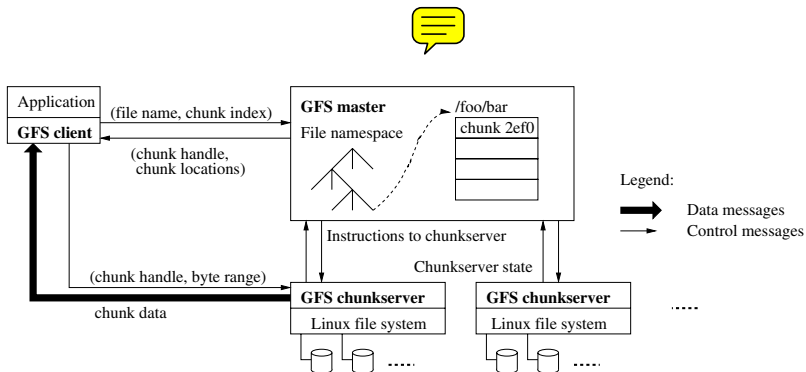
HDFS v.s. Hops-HDFS v2 (PCC version)

## Solution for Hops-HDFS

- Optimistic Concurrency Control with Snapshot Isolation on Semantic Related Group
- Performance Increase Up to 70 %
- Ensured by Passing 300+ Apache HDFS Unit Tests

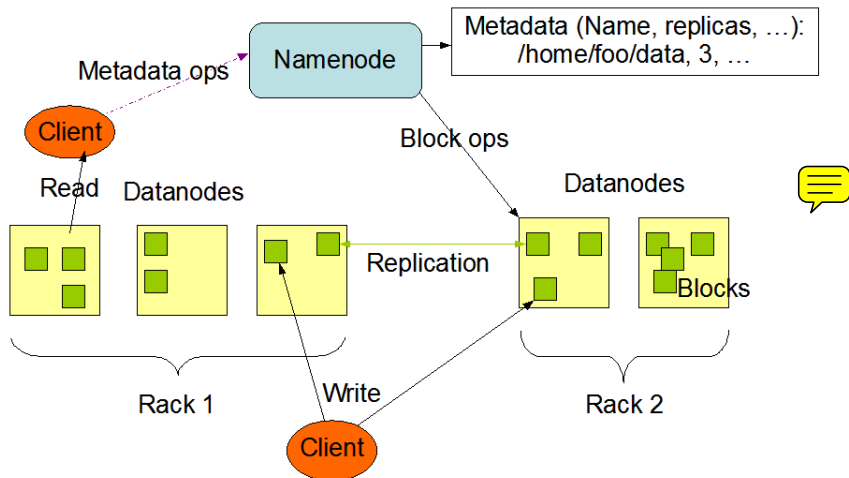


# GFS Architecture



# HDFS Architecture

## HDFS Architecture



# Isolation Level

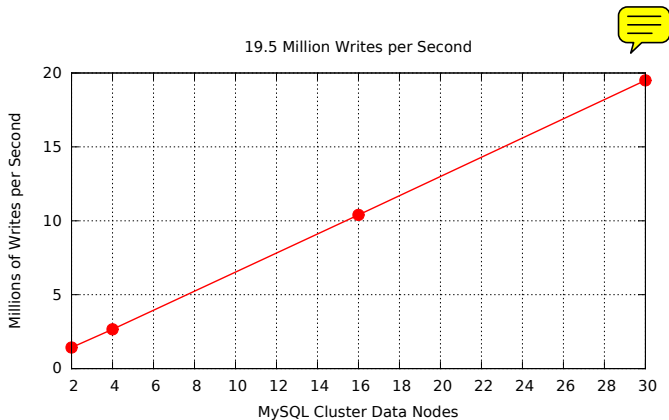
Berenson, Hal, et al. "A Critique of ANSI SQL Isolation Levels."  
ACM SIGMOD Record 24.2 (1995): 1-10.

Isolation Level	Lost Up-date	Fuzzy Read	Phantom	Read Skew	Write Skew
Read Uncommitted	✓	✓	✓	✓	✓
Read Committed	✓	✓	✓	✓	✓
Cursor Stability	some-times	some-times	✓	✓	some-times
Repeatable Read	X	X	✓	X	X
<b>Snapshot</b>	X	X	sometimes	X	✓
Serializable	X	X	X	X	X



# MySQL Cluster

- Distributed, in-memory, replicated database
- Supports only **Read Committed**
- High throughput:



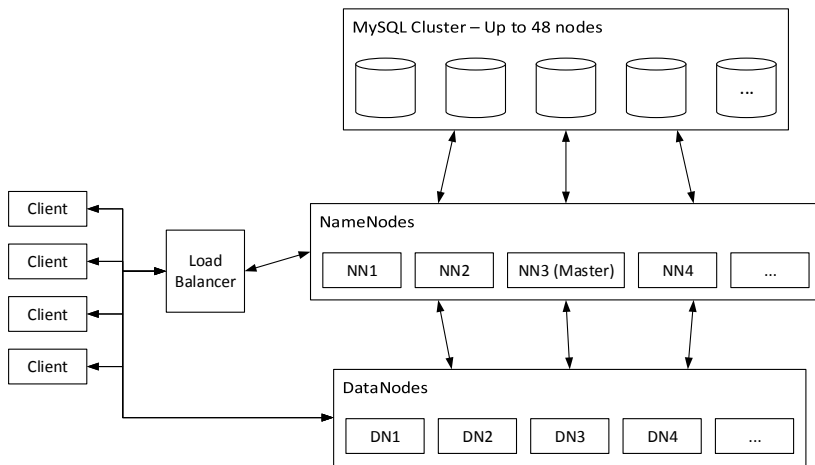


# Hops-HDFS

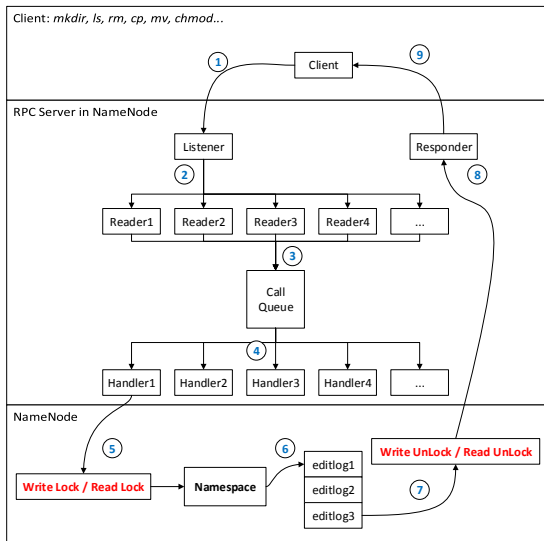
## Overcome Limitations in HDFS NameNode

- Scalability of the Namespace
- Throughput Problem
- Failure Recovery

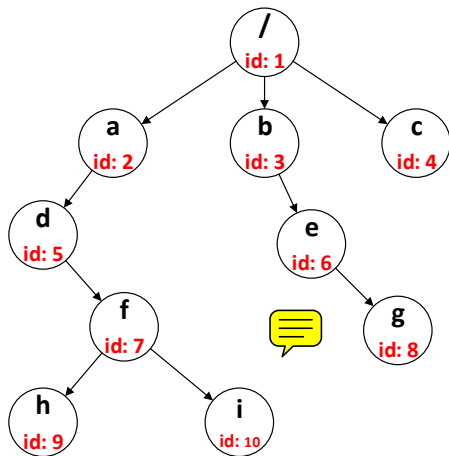
# Hops-HDFS Architecture



# Limitations in HDFS Namespace Concurrency Control



# Hops-HDFS Namespace Structure



id	parent_id	name
1	0	/
2	1	a
3	1	b
4	1	c
5	2	d
6	3	e
7	5	f
8	6	g
9	7	h
10	7	i

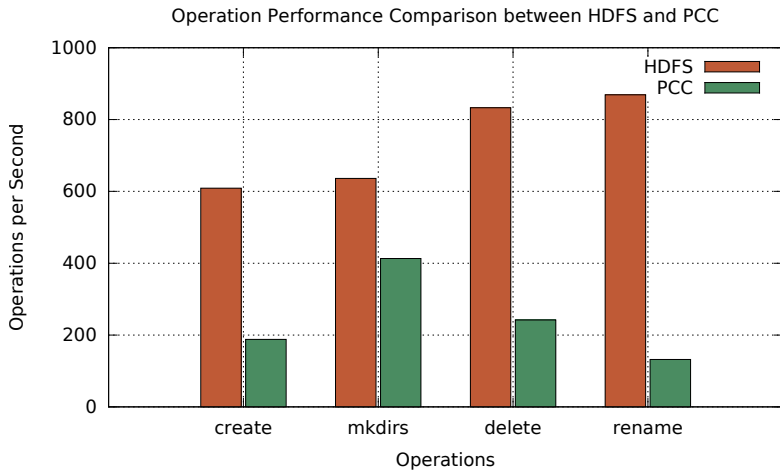
# Limitations in Hops-HDFS Namespace Concurrency Control

- Duplicated Round Trips
- Implicit Parent Locks:

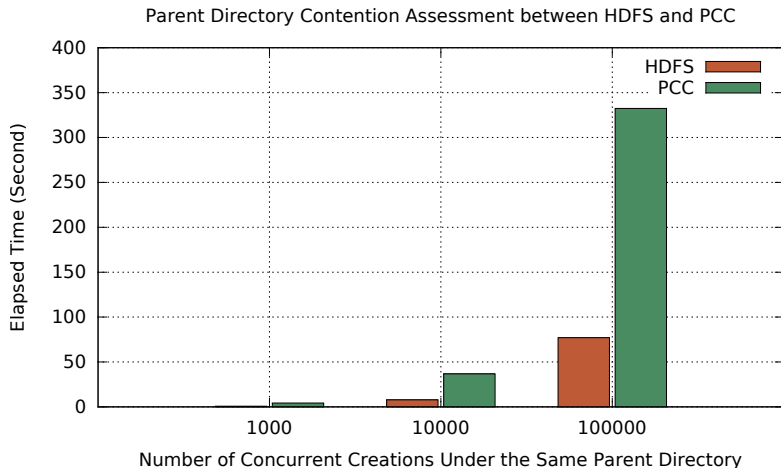


id	parent_id	name	Locks by Tx1	Locks by Tx2
1	0	/	R	R
2	1	a	R	R
3	1	b		
4	1	c		
5	2	d	R	R
6	3	e		
7	5	f	W	W (Block)
8	6	g		
9	7	h (Tx1)	W (Implicit)	W (Implicit) (Block)
10	7	i (Tx2)	W (Implicit)	W (Implicit) (Block)

# NameNode Throughput Benchmark



# Parent Directory Contention Assessment



# Resolving the Semantic Related Group

Path: /a/d/f/h

h: {/->a->d->f}

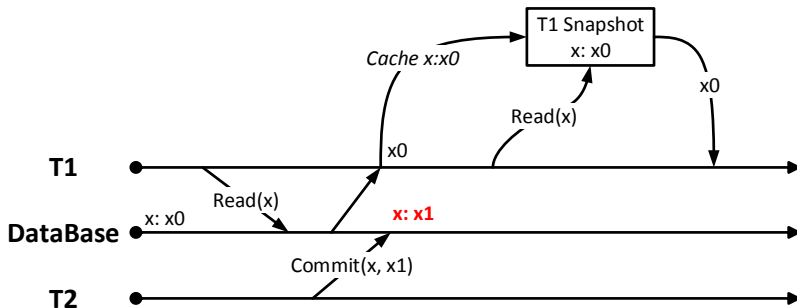
	id	parent_id	name	other parameters...
Related *	1	0	/	...
Related *	2	1	a	...
	3	1	b	...
	4	1	c	...
Related *	5	2	d	...
	6	3	e	...
Related *	7	5	f	...
	8	6	g	...
Selected ✓	9	7	h	...
	10	7	i	...



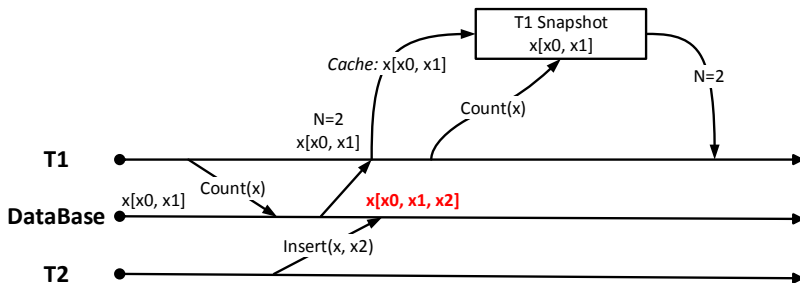
# Per-Transaction Snapshot Isolation

- Snapshot the whole Semantic Related Group
- Transaction performs on its own snapshot
- Preclude: *Fuzzy Read & Phantom Read*

# Snapshot Isolation Precludes Fuzzy Read



# Snapshot Isolation with Semantic Related Group Precludes Phantom Read



# Lock Mode in MySQL Cluster



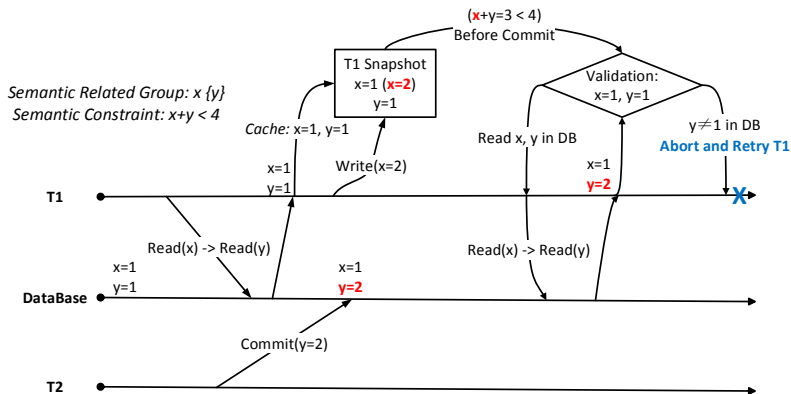
*Read\_Committed*: Consistent nonlocking reads (based on MVCC)

Lock Type	Shared	Exclusive	Read_Committed
Shared	✓	Block	✓
Exclusive	Block	Block	✓
Read_Committed	✓	✓	✓


# Per-Transaction Snapshot Isolation

- Read Phase: *Read\_Committed* on snapshot
- Validation Phase: *Shared* on related Rows, *Exclusive* on modified rows / Compare versions / Abort and retry
- Preclude: *Write Skew*

# OCC with Snapshot Isolation on Semantic Related Group Precludes Write Skew



# Total Order Update, Abort & Retry, and Version Increase

- Total order update modified rows by *ids*
- Abort and retry transactions if "new" rows already exist 
- Increase versions for successful update

# Four Phases in Algorithm

- Read Phase
- Execution Phase
- Validation Phase
- Update Phase





# Experimental Testbed

## MySQL Cluster

6 data nodes, 1 Gbps LAN, Intel Xeon X5660 CPU @ 2.80GHz,  
 $6 \times 6 = 36$  GB RAM, 2 data replicas

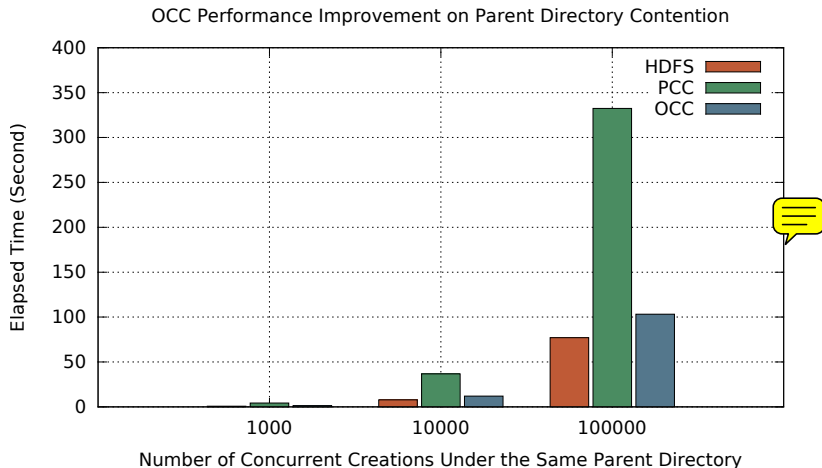
## NameNode and Clients

Intel i7-4770T CPU @ 2.50GHz and 16 GB RAM

## MySQL Cluster and NameNode

100 Mbps LAN

# Parent Directory Contention Assessment



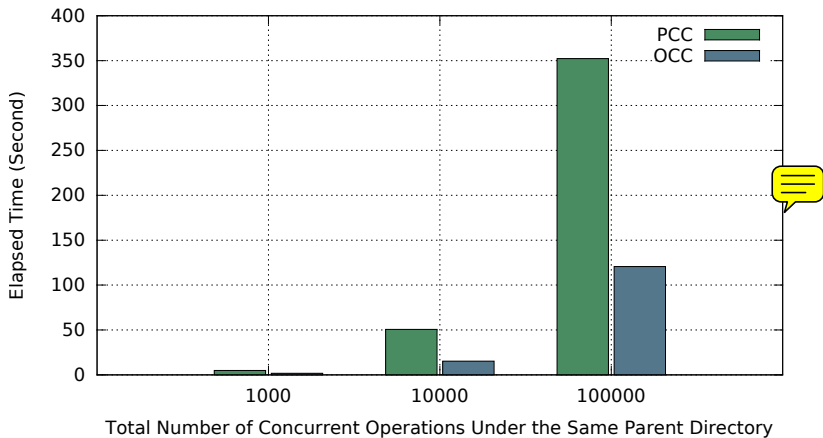
# Parent Directory Contention Assessment

<b>Num. of Concurrent Creation</b>	<b>1000</b>	<b>10000</b>	<b>100000</b>
HDFS	0.82s	7.83s	77.13s
PCC	4.35s	36.74s	332.36s
OCC	1.36s	12.01s	103.23s
PCC / HDFS	530.5%	469.2%	430.9%
OCC / HDFS	165.9%	153.4%	133.8%
<b>OCC Improvement: (PCC-OCC) / PCC</b>	<b>68.7%</b>	<b>67.3%</b>	<b>68.9%</b>



# Read-Write Mixed Workload

OCC Performance Improvement on Read-Write Mixed Workload



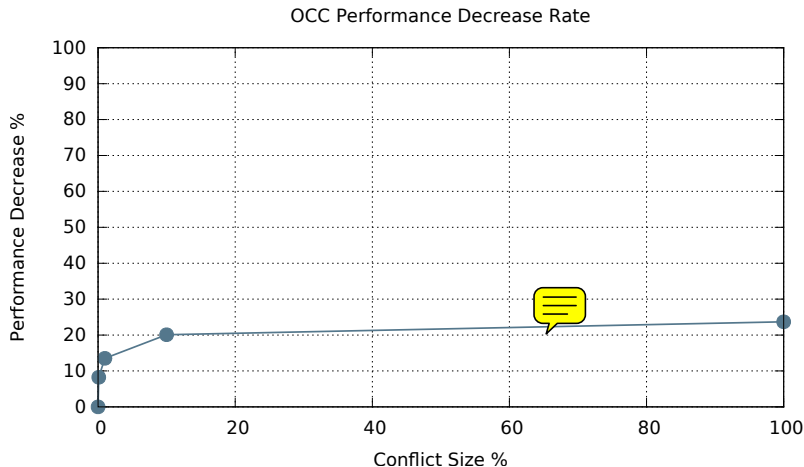
# Read-Write Mixed Workload

<b>Concurrent Read+Creation</b>	<b>1000</b>	<b>10000</b>	<b>100000</b>
PCC	4.92s	50.69s	352.25s
OCC	1.78s	15.31s	120.64s
<b>OCC Improvement: (PCC-OCC) / PCC</b>	<b>63.8%</b>	<b>69.8%</b>	<b>65.8%</b>

# OCC Performance with Different Size of Conflicts

<b>Creations for 10000 Operations</b>	<b>Conflict Size</b>	<b>Elapsed Time (Second)</b>	<b>Performance Decrease</b>
1	100%	14.53	23.7%
10	10%	14.11	20.1%
100	1%	13.51	15.0%
1000	0.1%	12.72	8.23%
10000	0%	11.75	0%

# OCC Performance Decrease Rate



# Implementation Correctness Assessment

Ensured by passing 300+ Apache HDFS Unit Tests



# Conclusion & Future Work

## Conclusion

- Increase Performance up to 70 %
- Maintain HDFS Strong Consistency Semantics

## Future Work

- OCC implementation on other operations
- OCC evaluation on multiple NameNodes

Thank you!