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
100 million	**60 GB**	**60 PB**
1 billion	600 GB	600 PB

Hop-HDFS and Its Limitation

Distributed NameNode Architecture

Restricted Concurrency

Problem Statement

HDFS

System-level Lock

Hop-HDFS v1

System-level Lock

Hop-HDFS v2

Row-level Lock

MySQL Cluster

Read Committed / Anomalies

Contribution

Architectures and Namespace Concurrency Control

GFS, HDFS, Hop-HDFS and MySQL Cluster

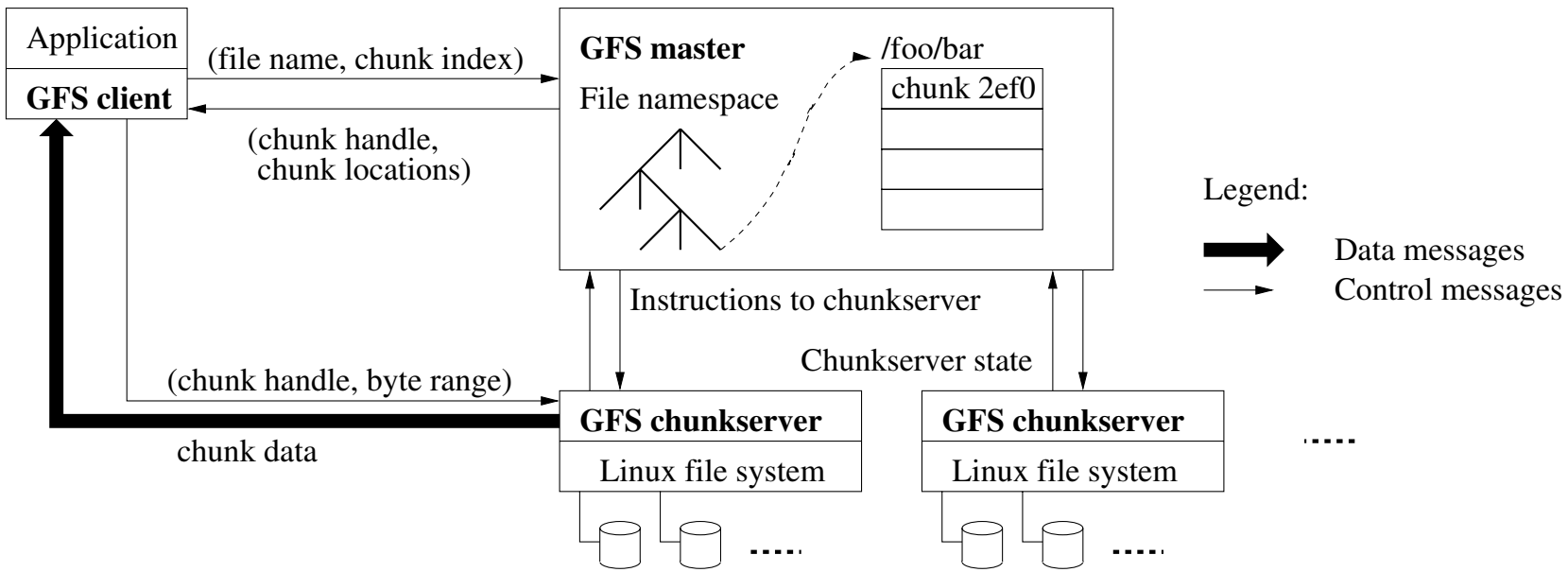
Performance Accessment and Limitation Analysis

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

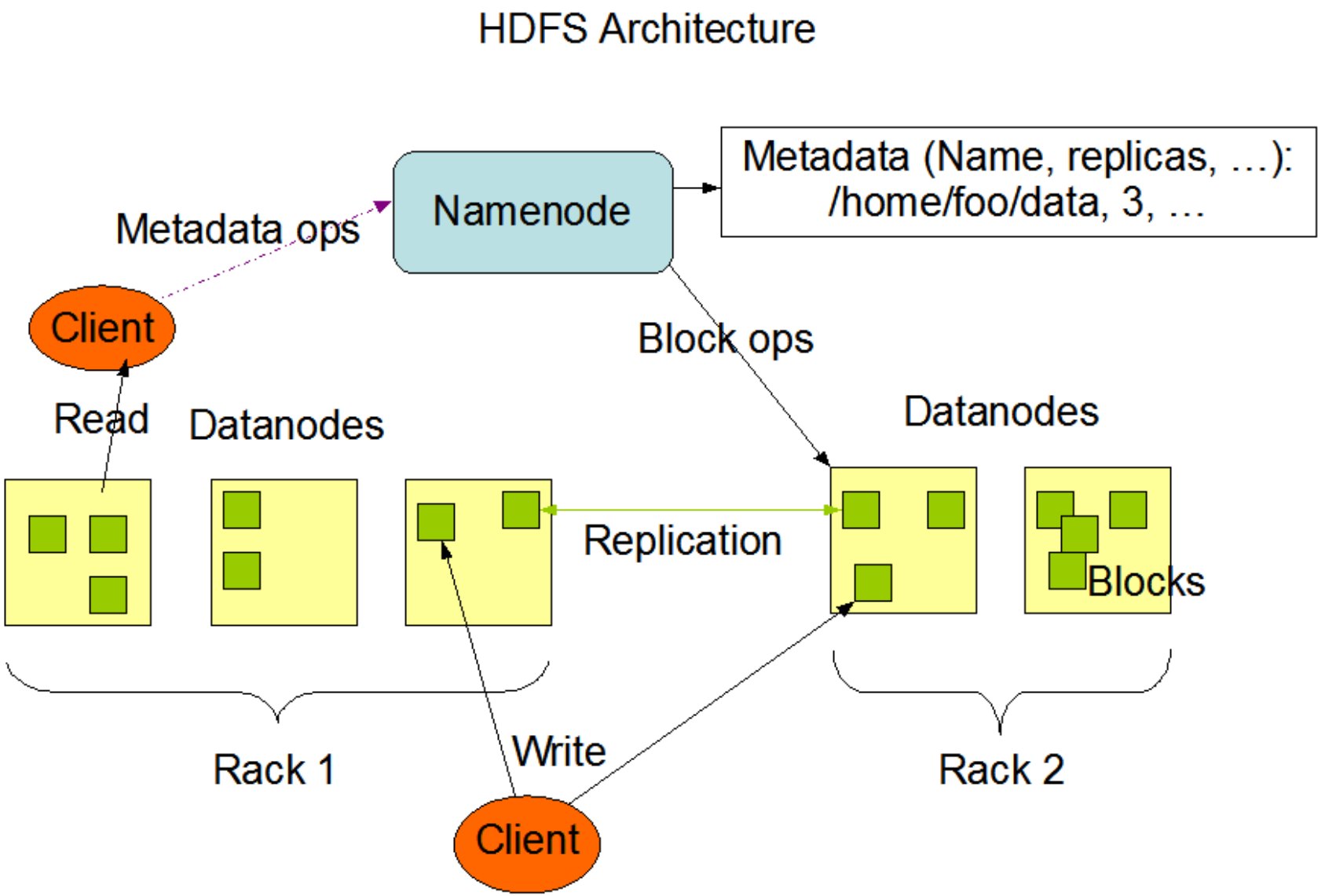
Solution for Hop-HDFS

Optimistic Concurrency Control with Snapshot Isolation on Semantic Related Group

GFS Architecture



HDFS Architecture



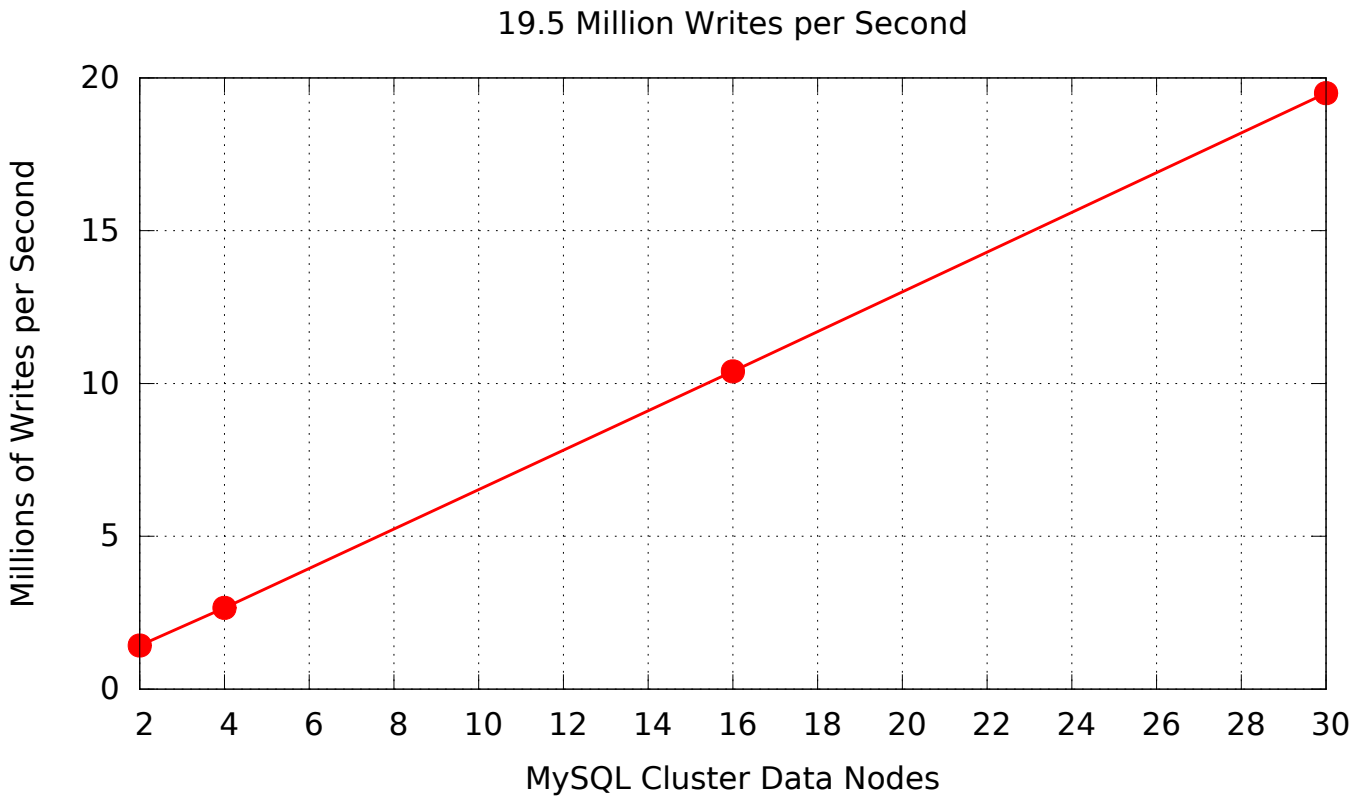
Isolation Level

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

Isolation Level	**Lost Update**	**Fuzzy Read**	**Phantom**	**Read Skew**	**Write Skew**
Read Uncommitted	✓	✓	✓	✓	✓
Read Committed	✓	✓	✓	✓	✓
Cursor Stability	sometimes	sometimes	✓	✓	sometimes
Repeatable Read	X	X	✓	X	X
Snapshot	X	X	sometimes	X	✓
Serializable	X	X	X	X	X

MySQL Cluster

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

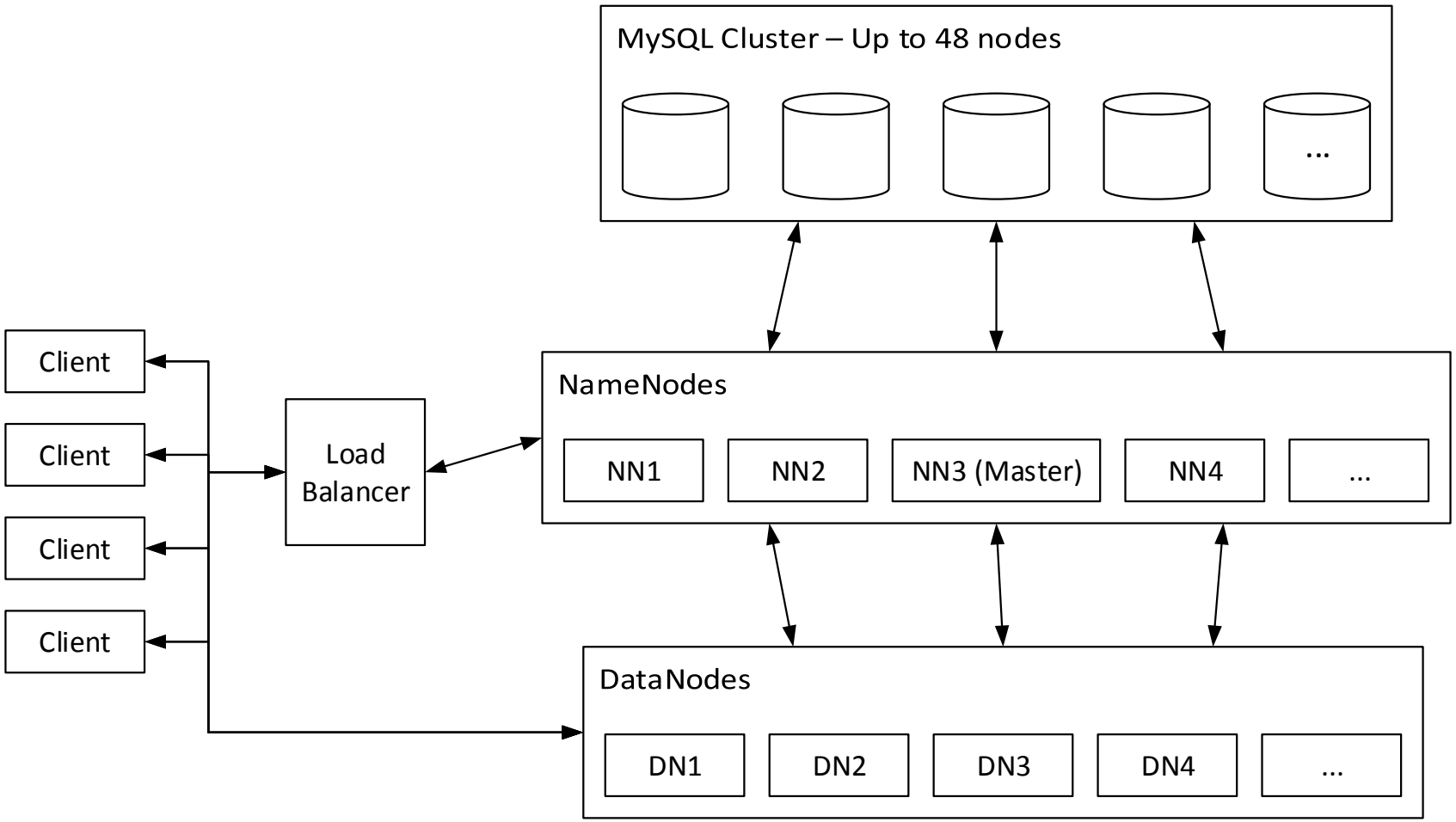


Hop-HDFS

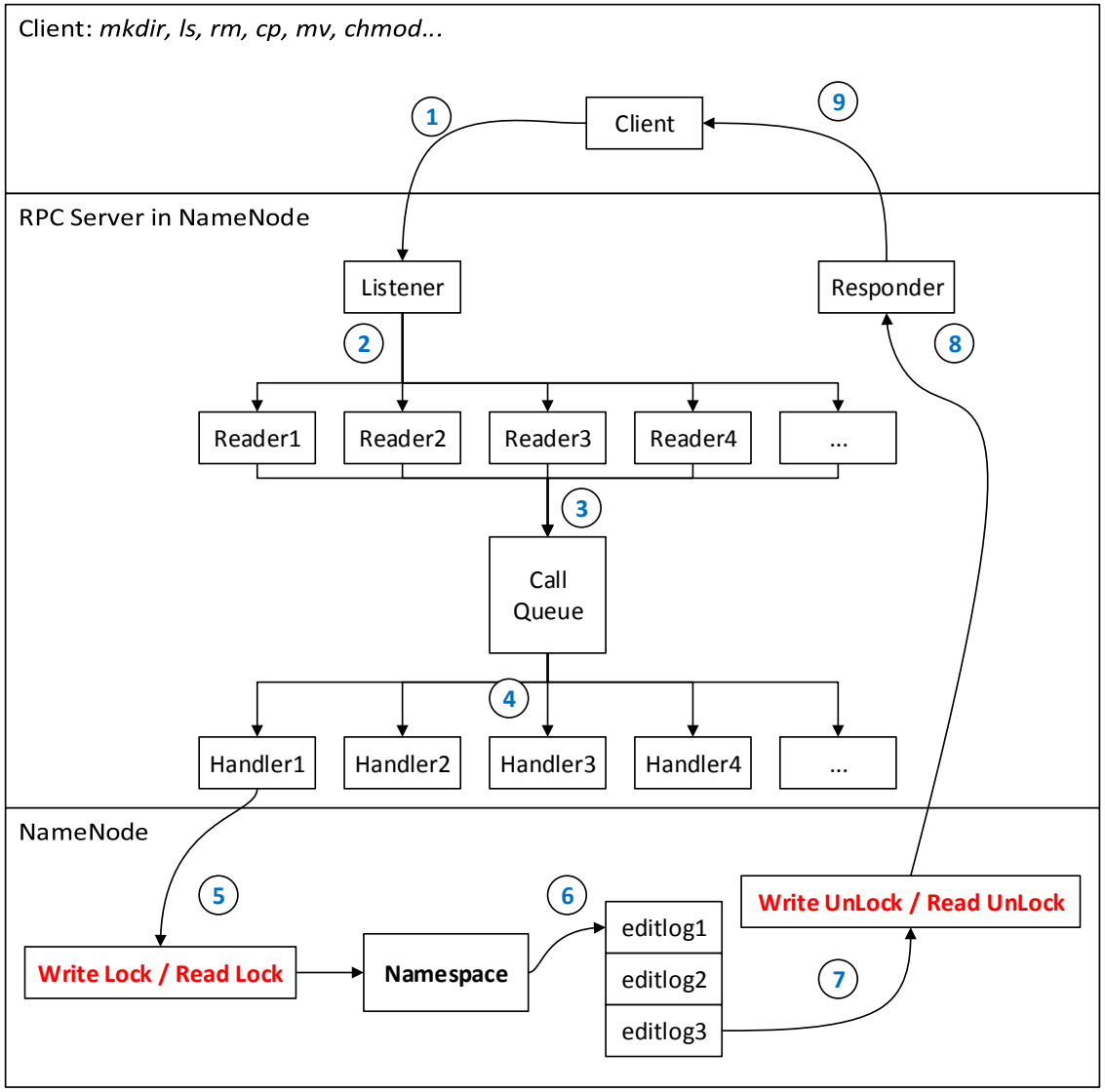
Overcome Limitations in HDFS NameNode

- Scalability of the Namespace
- Throughput Problem
- Failure Recovery

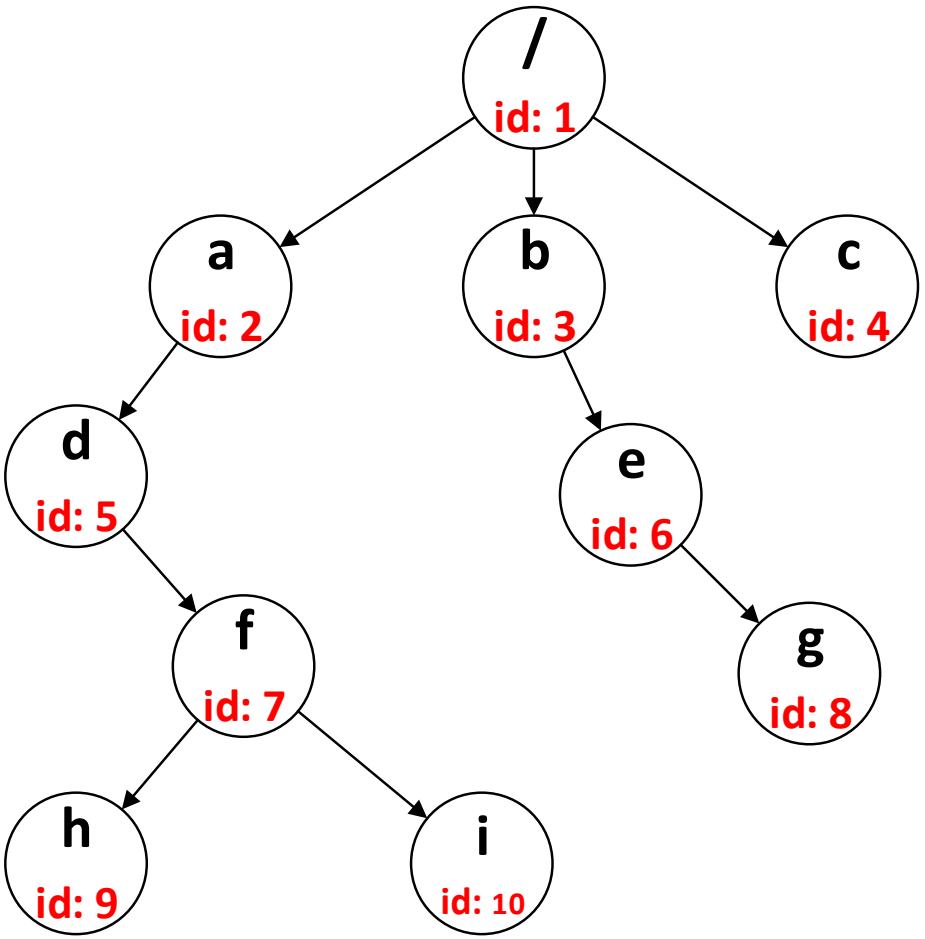
Hop-HDFS Architecture



Limitations in HDFS Namespace Concurrency Control



HDFS Namespace Structure



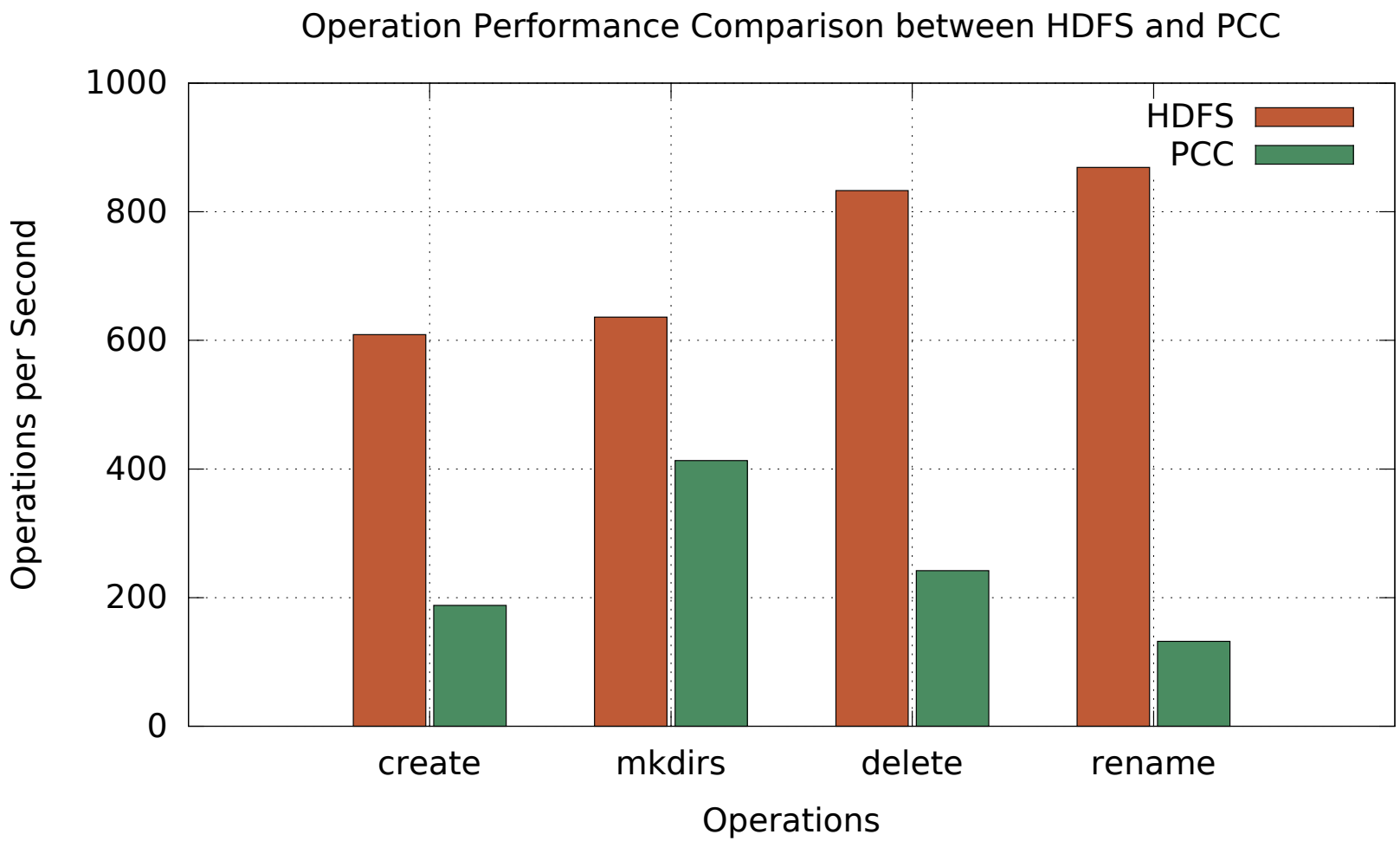
id	parent_id	name
1	0	/
2	1	a
3	1	b
4	1	c
5	2	d
6	3	e
7	5	f
8	6	g
9	7	h
10	7	i

Limitations in Hop-HDFS Namespace Concurrency Control

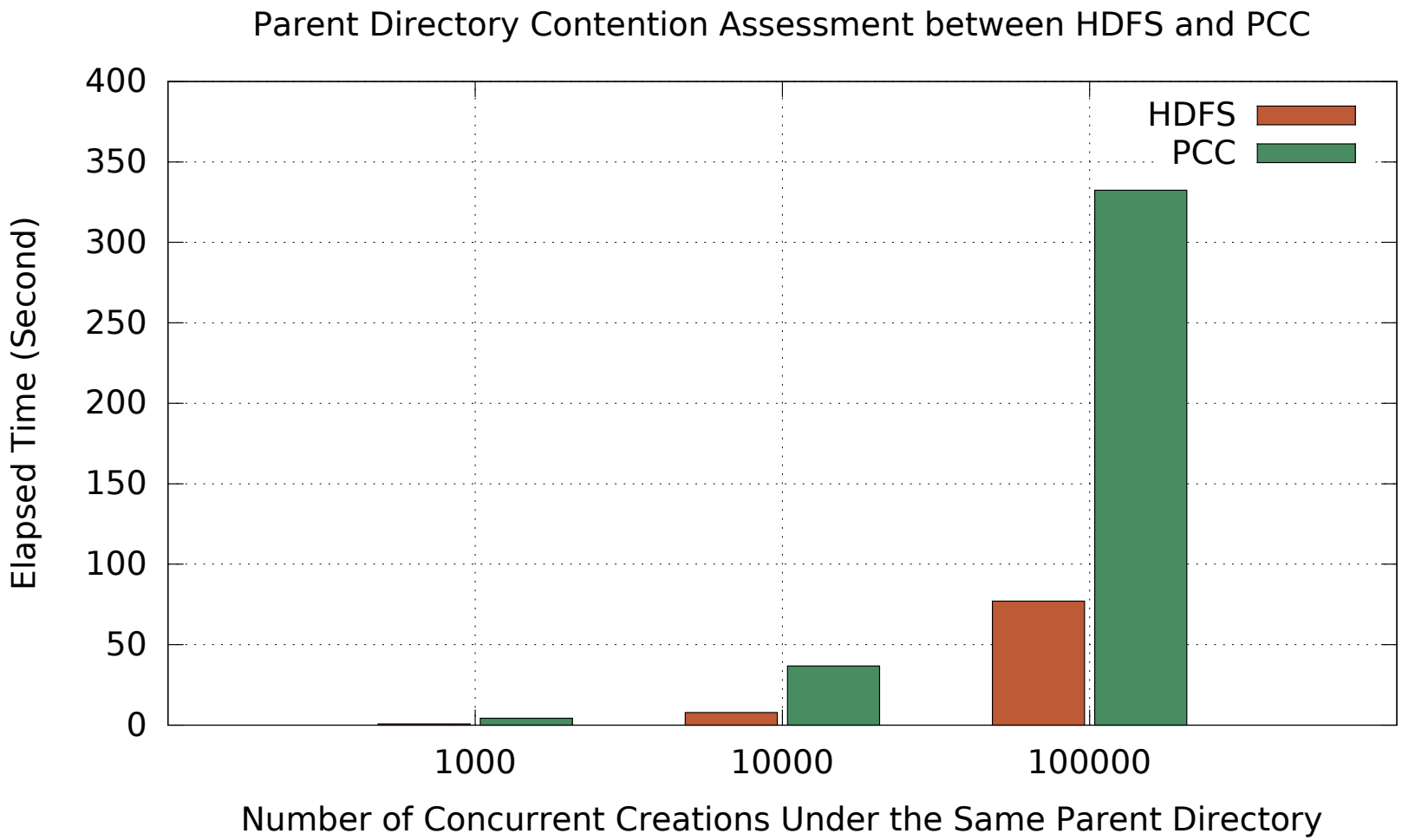
- Duplicated Round Trips
- Implicit Parent Locks:

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

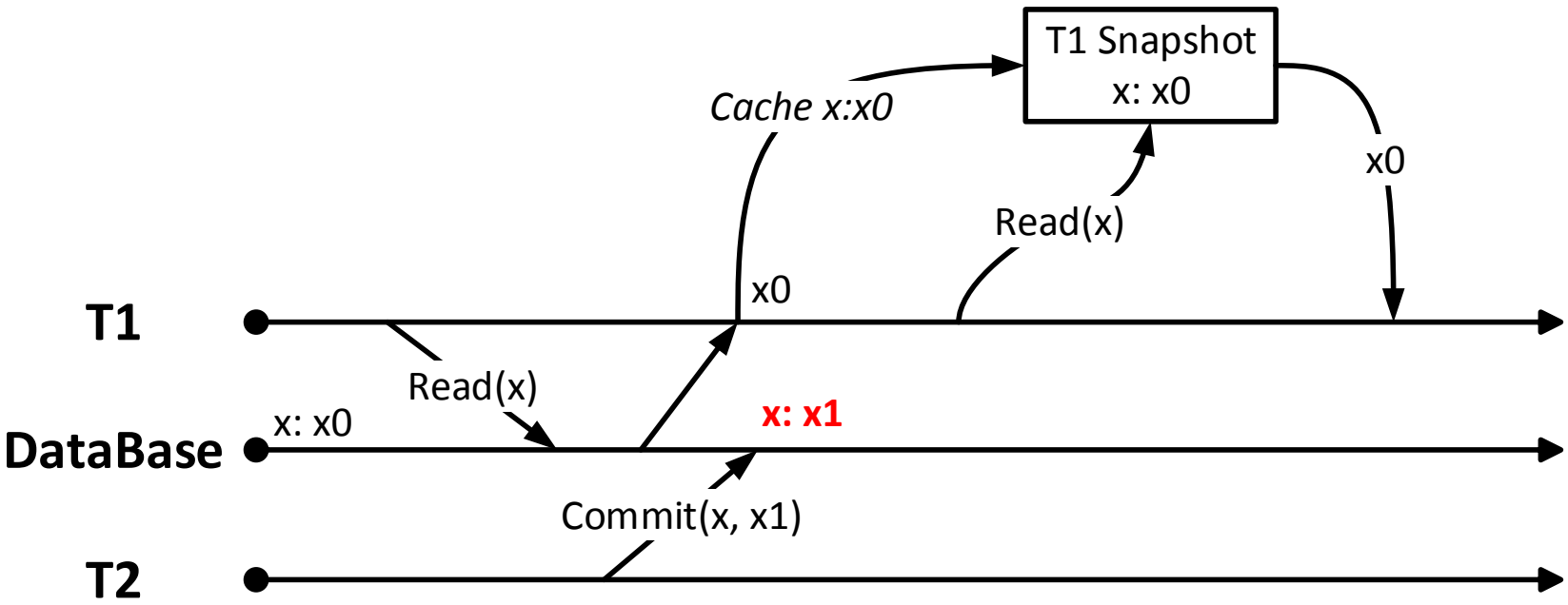
h: {/->a->d->f}

	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	...

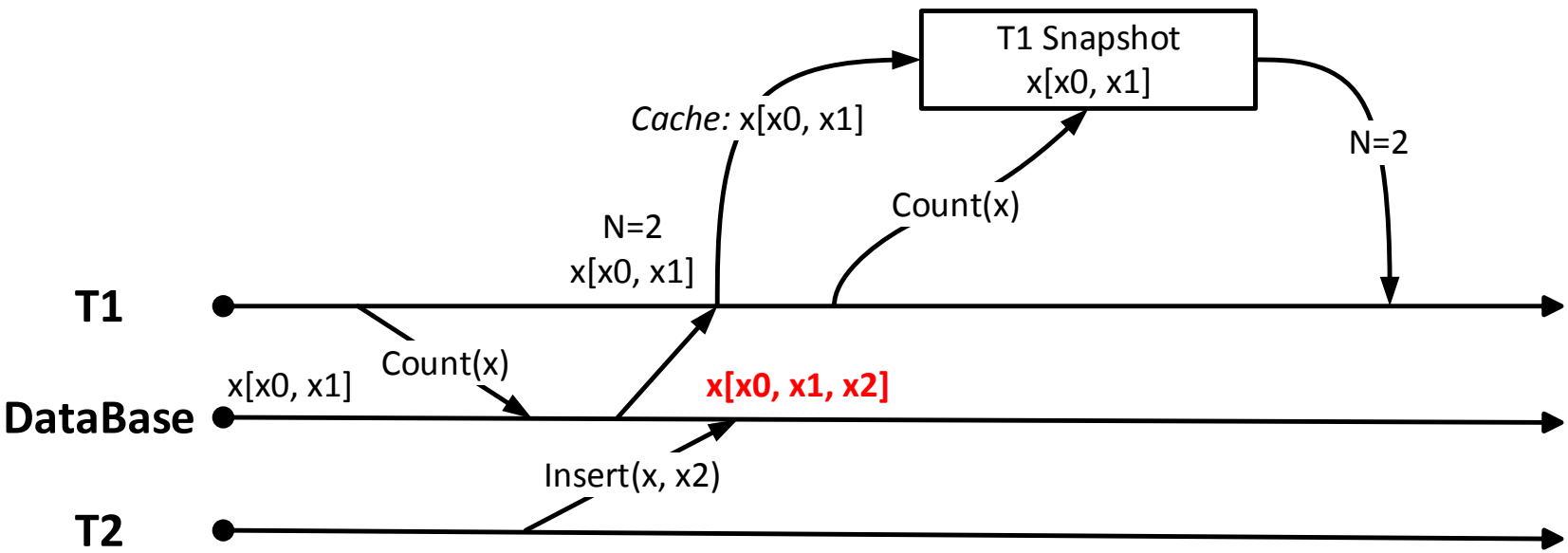
Per-Transaction Snapshot Isolation

- 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

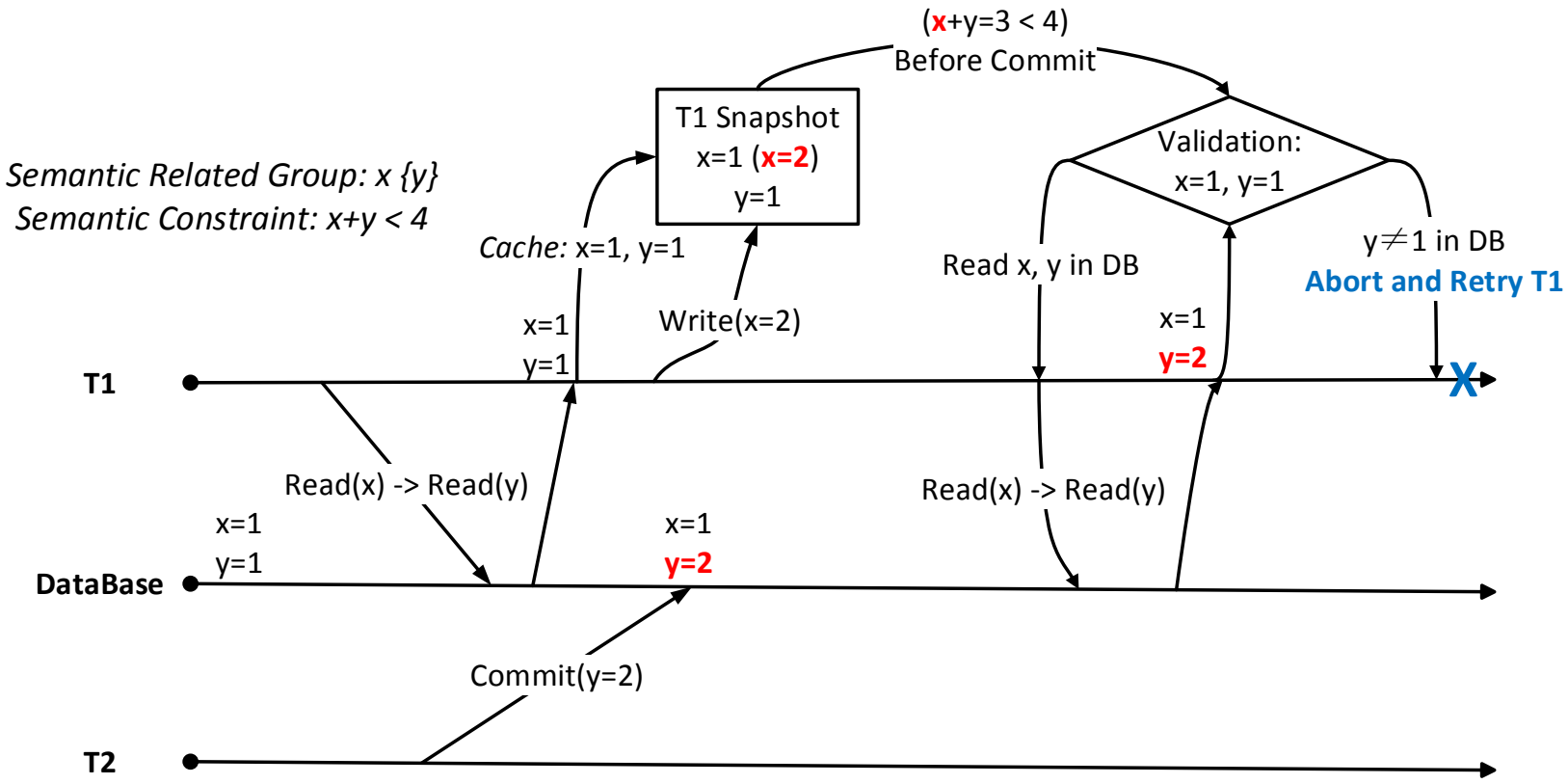
Read_Committed: Consistent nonlocking reads (based on MVCC)

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



Per-Transaction Snapshot Isolation

- Total order update modified rows by *ids*
- Abort and retry transactions if "new" rows already exists
- 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*6=36 GB RAM, 2 data replicas

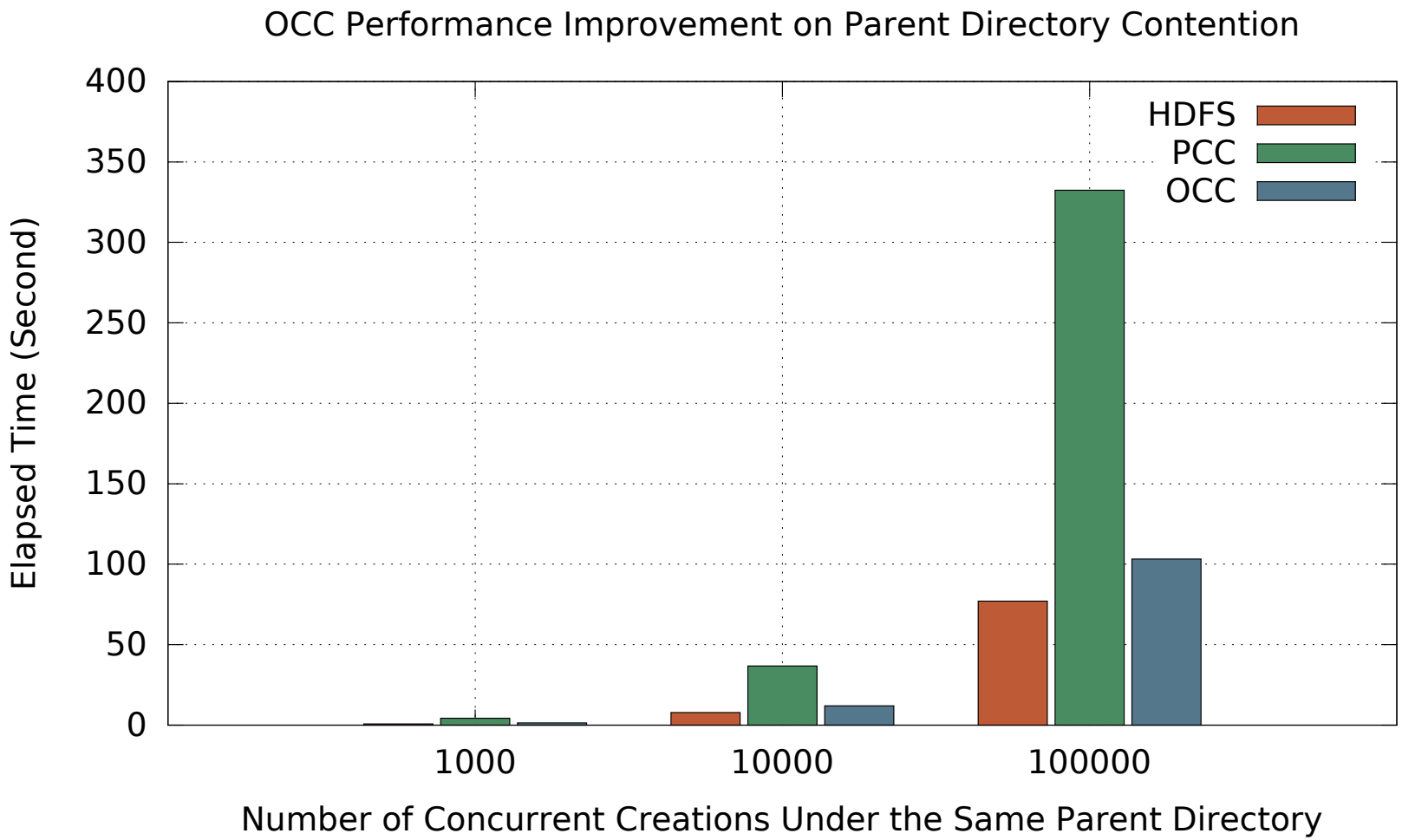
NameNode and Clients

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

MySQL Cluster and NameNode

100 Mbps LAN

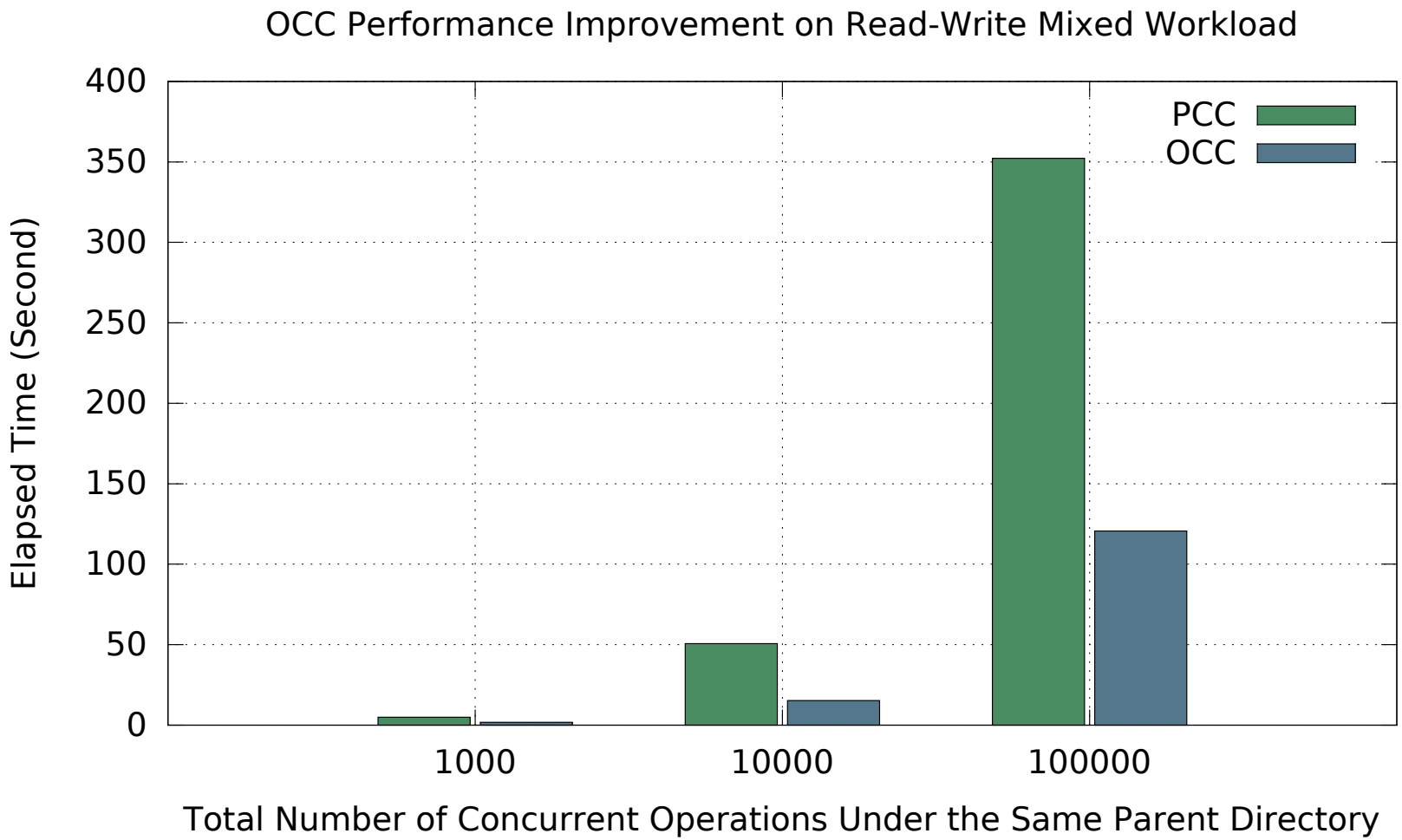
Parent Directory Contention Assessment



Parent Directory Contention Assessment

Num. of Concurrent Creation	**1000**	**10000**	**100000**
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%
OCC Improvement: (PCC-OCC) / PCC	**68.7%**	**67.3%**	**68.9%**

Read-Write Mixed Workload



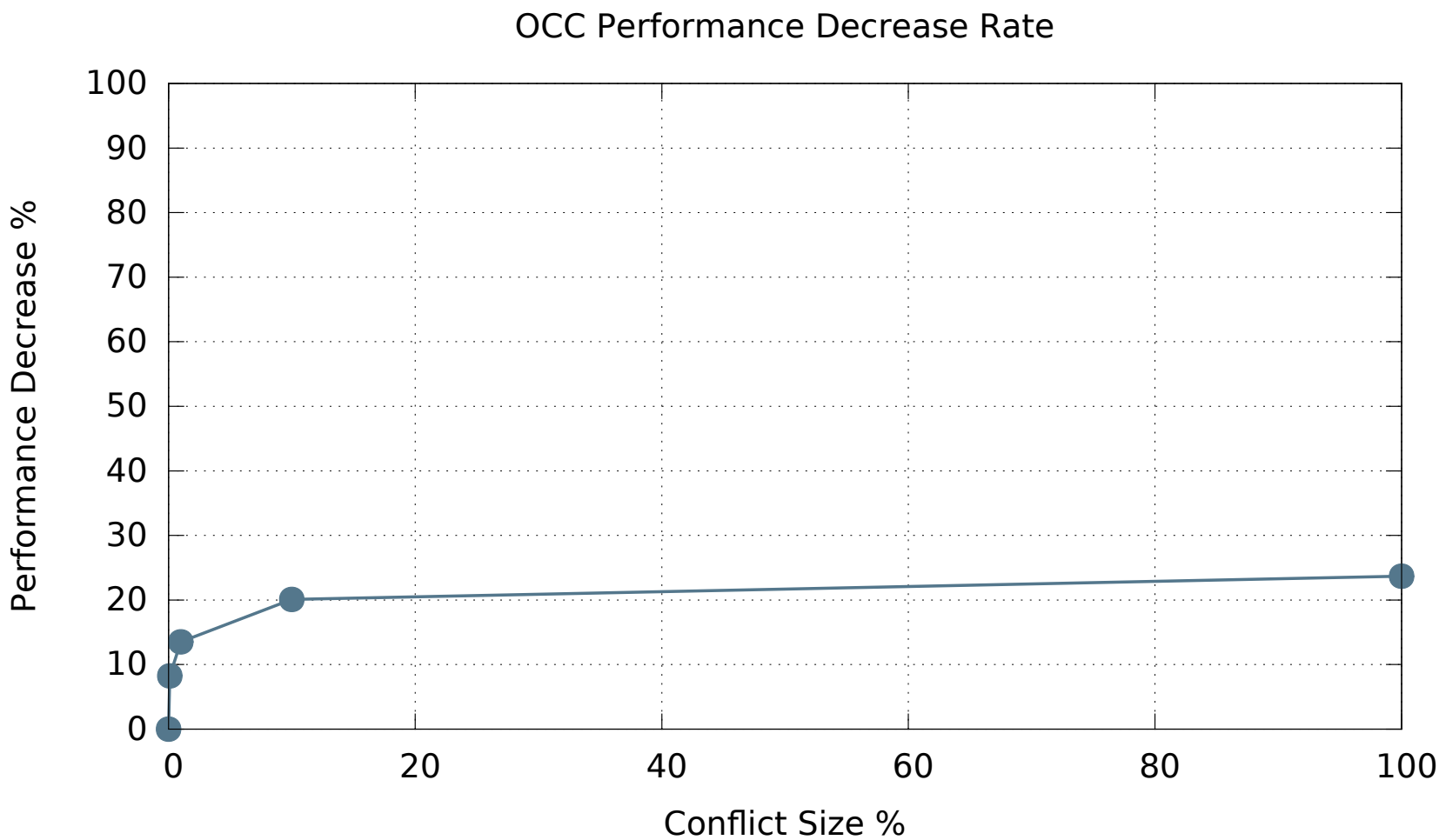
Read-Write Mixed Workload

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

OCC Performance with Different Size of Conflicts

Creations for 10000 Operations	Conflict Size	Elapsed Time (Second)	Performance Decrease
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



Conclusion & Future Work

Conclusion

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

Future Work

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

Thank you!