Speeding up the Inter-Planetary File System (IPFS)

Semester project presentation

Project Advisor: Prof. Bryan Ford

Project Supervisor: Cristina Basescu

Student: Guillaume Michel

Laboratory: DEDIS

IPFS

- Content addressed distributed peer-to-peer filesystem
- Developed by *Protocol Labs*
- Combining ideas from *Git, BitTorrent* & *Kademlia*

Example

$ ipfs add cat.gif

added QmXTqCnyGrg1ruC  lbo3Yq cat.gif

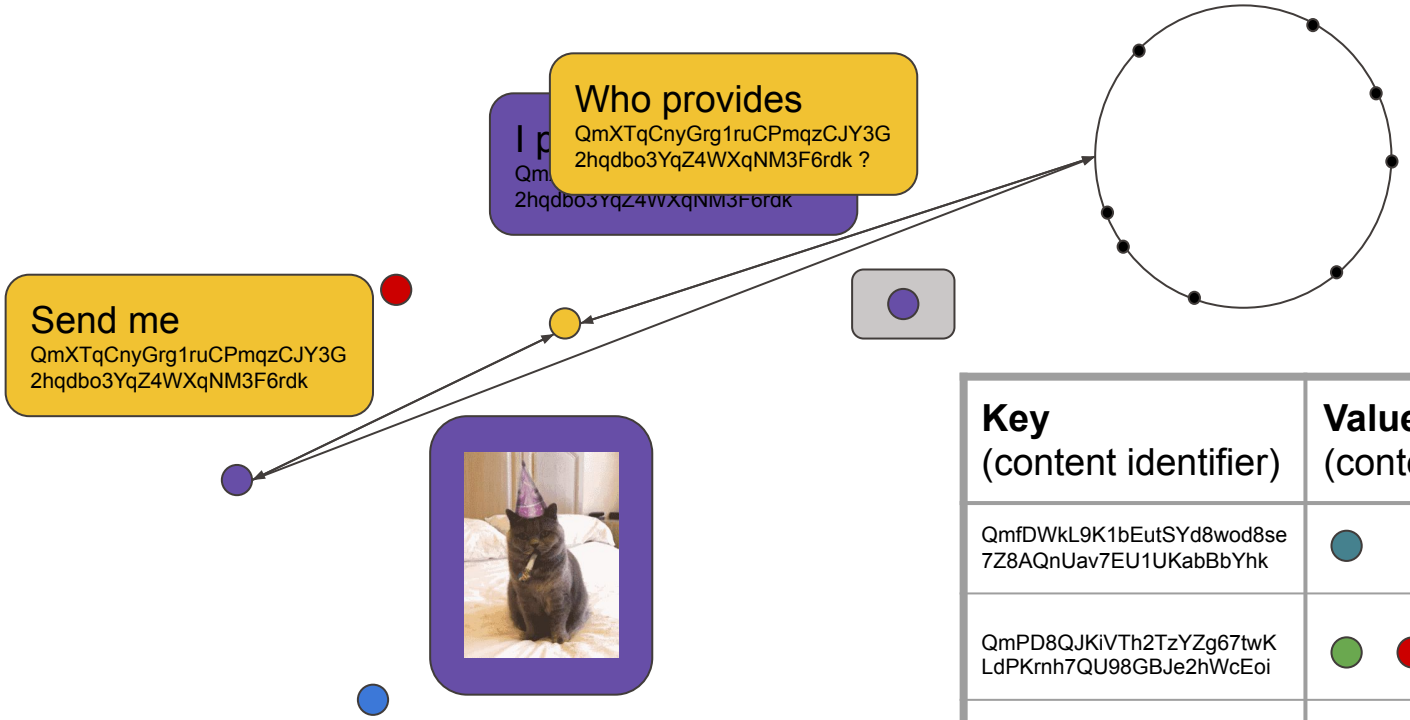
$ ipfs get QmXTqCnyGrg1r nqdbo3YqZ4WXqNM3F6rdk

Saving file(s) to QmXTqCn JY3G2hqdbo3YqZ4WXqNM3F6rdk

$ open QmXTqCnyGrg1ruCPmqzCJY3G2hqdbo3YqZ4WXqNM3F6rdk

Example

IPFS DSHT



Key (content identifier)	**Value** (content provider)
QmfDWkL9K1bEutSYd8wod8se 7Z8AQnUav7EU1UKabBbYhk	
QmPD8QJKiVTh2TzYZg67twK LdPKrnh7QU98GBJe2hWcEoi	

Goal of the project

- Reduce the pair interaction latency in IPFS

Pair interaction latency in IPFS:

For a pair of nodes (W; R), the pair interaction latency is the time from the moment W starts to write a file to IPFS until R has fetched it.

Agenda

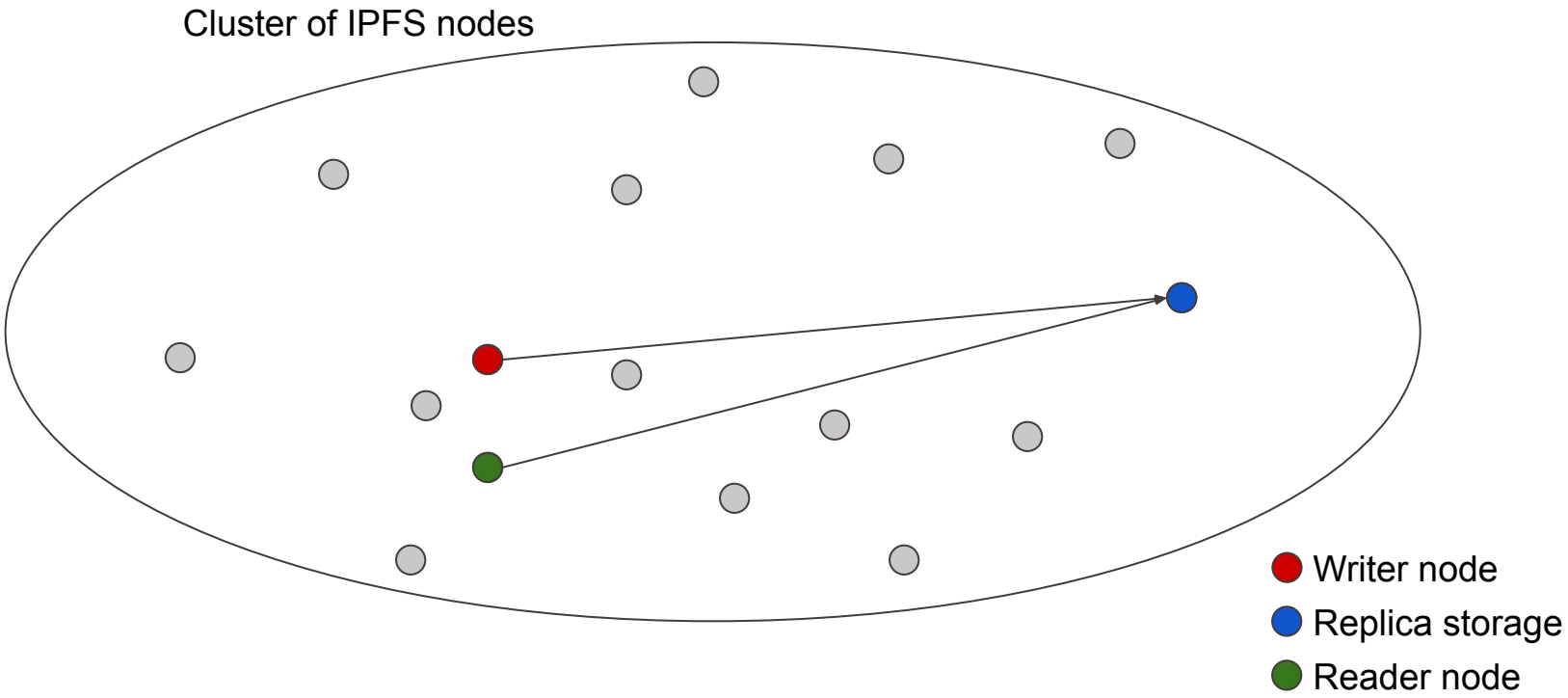
1. IPFS introduction
2. **Crux**
3. Cruxifying IPFS
4. Vanilla vs Cruxified IPFS performance
5. Conclusion

Crux

- Enhances locality in existing distributed systems
- Replicate data at *smart* locations
- Upper bound to worst-case latency of any pair of nodes in a network:

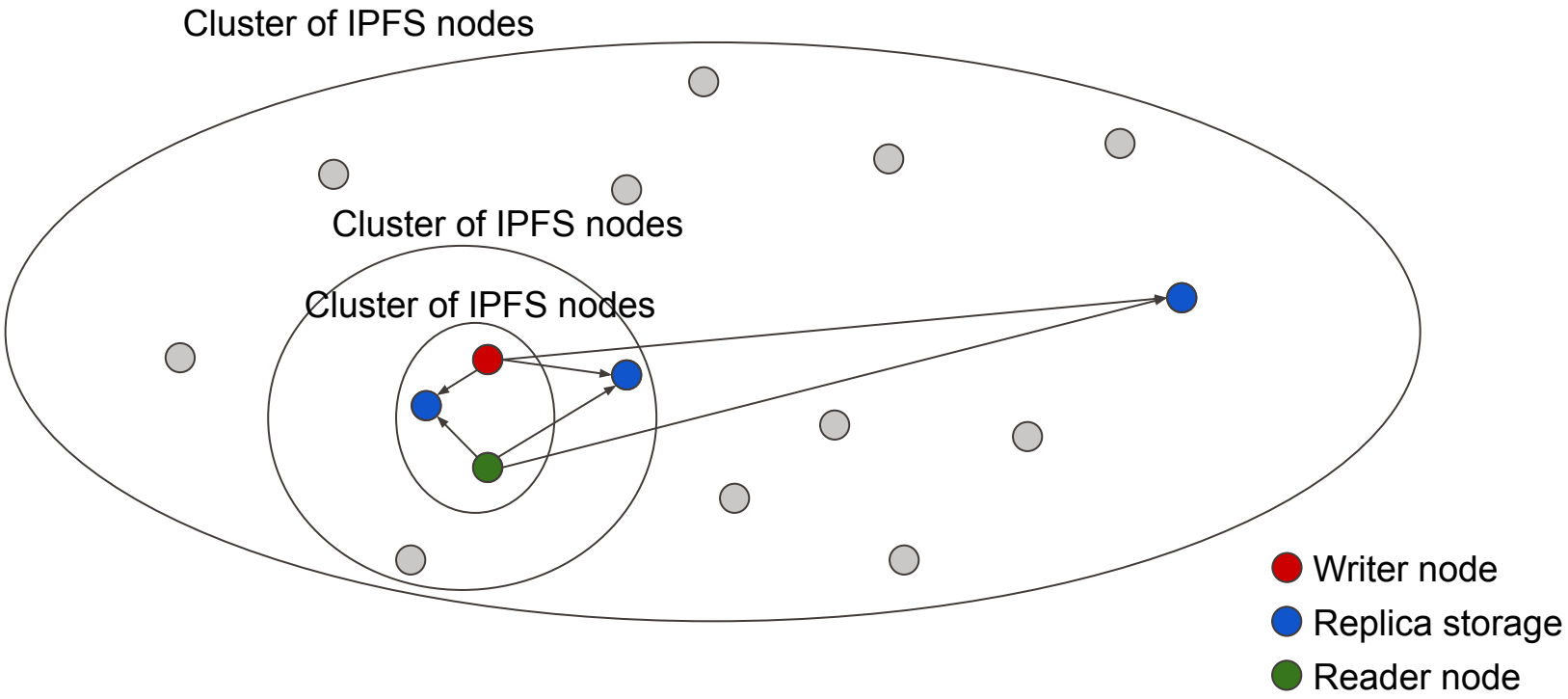
  Small multiple of their network latency (RTT)

Vanilla IPFS deployment



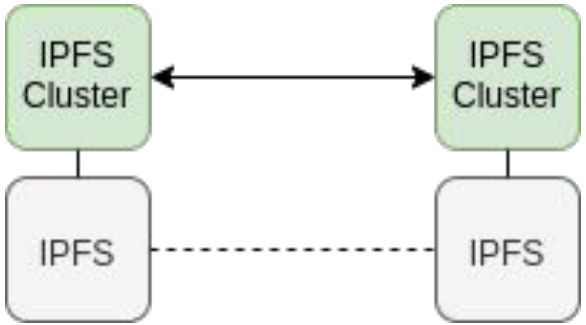
Guillaume Michel

Cruxified IPFS deployment



IPFS Cluster

- IPFS Cluster handles replication management
- Runs on top of IPFS
- Interact through IPFS API



IPFS Cluster
Pinset & Consistency

- IPFS Cluster maintains a distributed global pinset

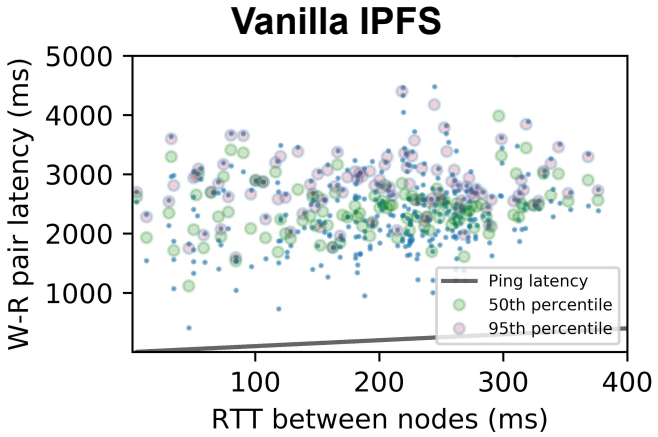
- This pinset keeps track for each file:
  - Location of all replicas
  - The last version

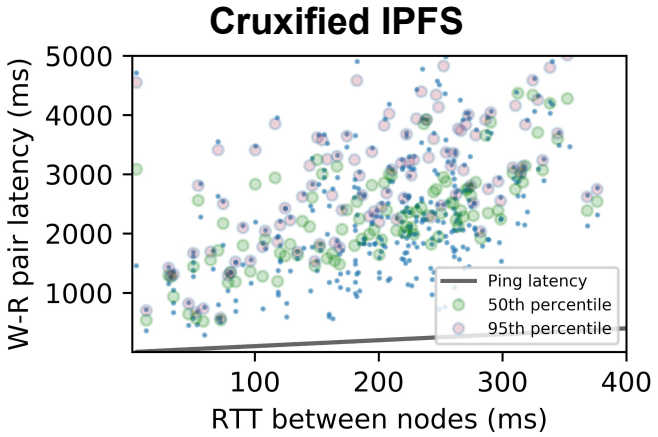
- 2 consistency components:
  - Raft (Strongly Consistent)
  - CRDT (Strongly Eventually Consistent)

Experiment

- Comparison of Vanilla & Cruxified IPFS systems
- Random topology of 20 nodes on *Deterlab*
- IPFS Cluster in CRDT consensus mode
- Replication factor: 2
- Crux: 3 levels of landmarks
- File size: 2KiB (single block)
- Machines specs: 16GiB RAM, Xeon E5-2420 v2 processor, 100Mib/s links
- Write + Read operations on 2000 random pairs of nodes

Write + Read pair latency

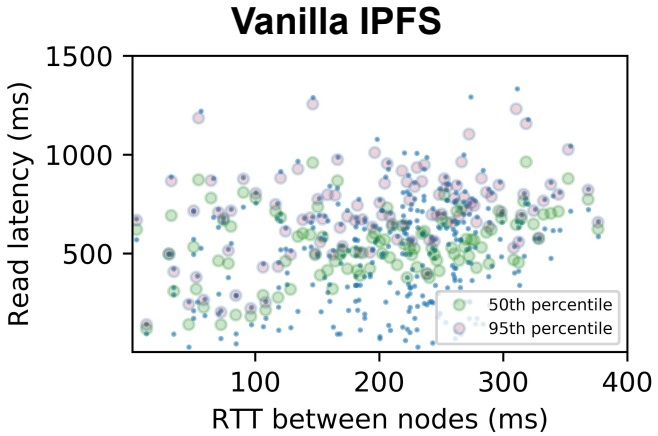


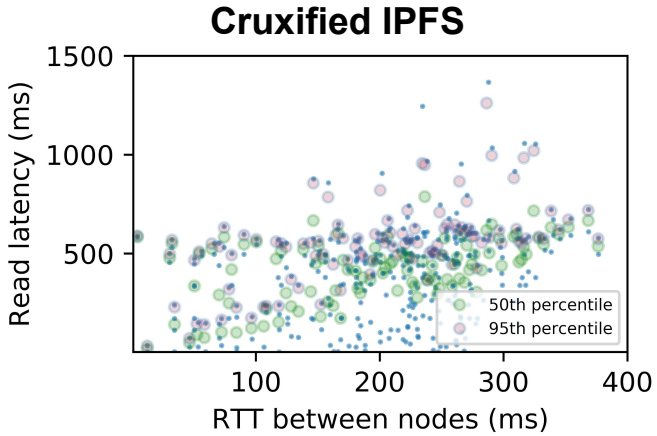


Average of Write + Read pair latency in ms according to the RTT between the nodes

	0-50 ms	50-100 ms	100-150 ms	150-200 ms	200-250 ms	250-300 ms	300+ ms
Vanilla IPFS	2032	2110	2213	2403	2439	2585	2775
Cruxified IPFS	1666	1890	2031	2392	2482	2740	3242
Improvement rate	18.0%	10.4%	8.2%	0.04%	-1.7%	-5.6%	-14.4%

Read latency

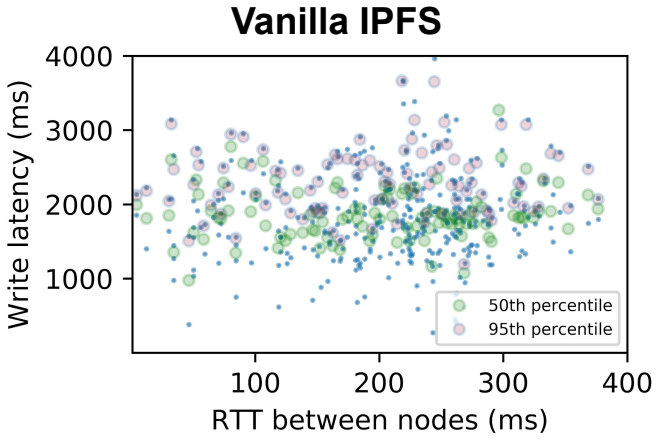


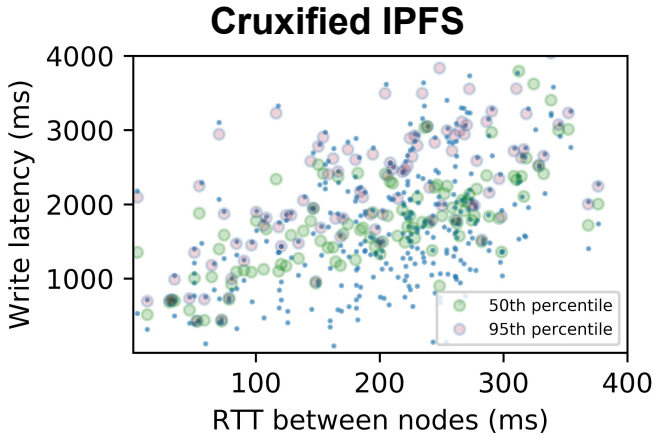


Average of Read pair latency in ms according to the RTT between the nodes

	0-50 ms	50-100 ms	100-150 ms	150-200 ms	200-250 ms	250-300 ms	300+ ms
Vanilla IPFS	382	452	492	557	571	611	683
Cruxified IPFS	254	323	359	431	438	497	593
Improvement rate	33.5%	28.5%	27.0%	22.6%	23.3%	18.5%	13.1%

Write latency





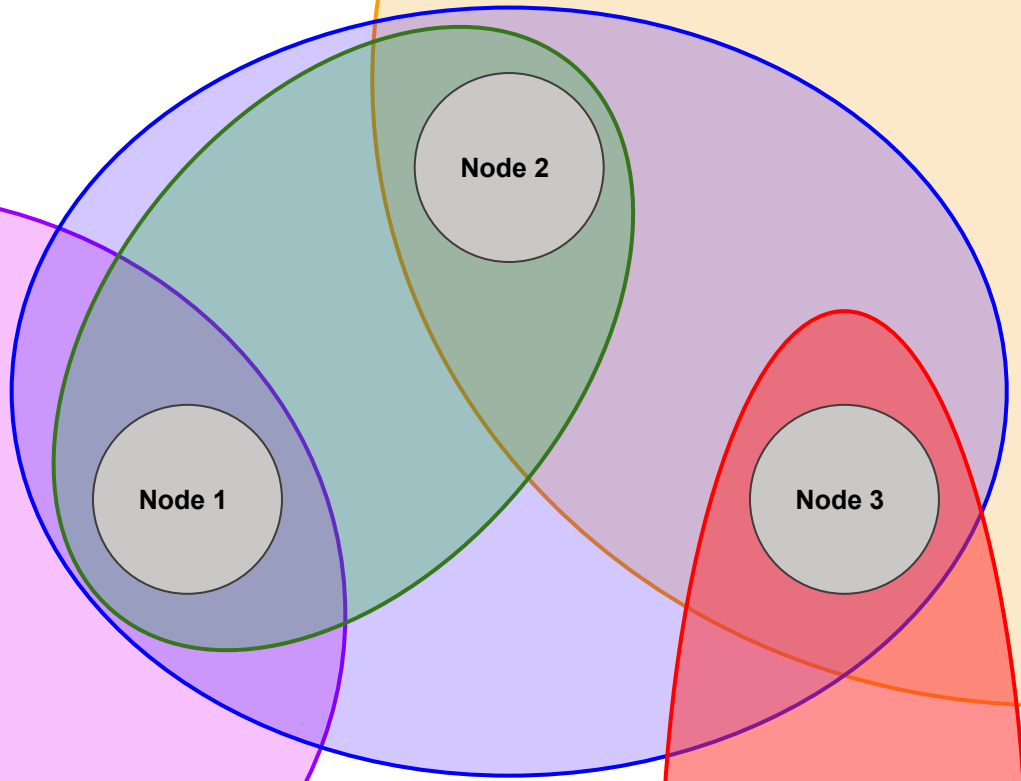
Average of Write pair latency in ms according to the RTT between the nodes

	0-50 ms	50-100 ms	100-150 ms	150-200 ms	200-250 ms	250-300 ms	300+ ms
Vanilla IPFS	1650	1658	1721	1846	1868	1974	2092
Cruxified IPFS	1412	1567	1672	1961	2044	2243	2649
Improvement rate	14.4%	5.5%	2.8%	-5.9%	-8.6%	-12.0%	-21.0%

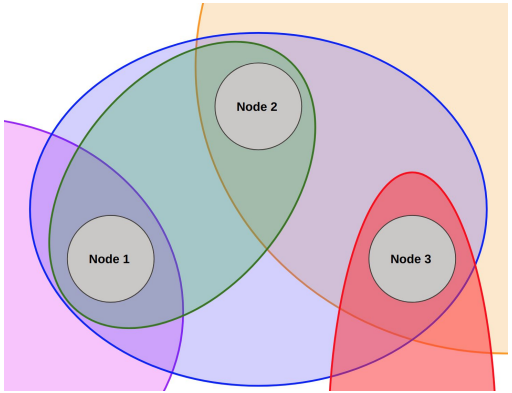
Conclusion

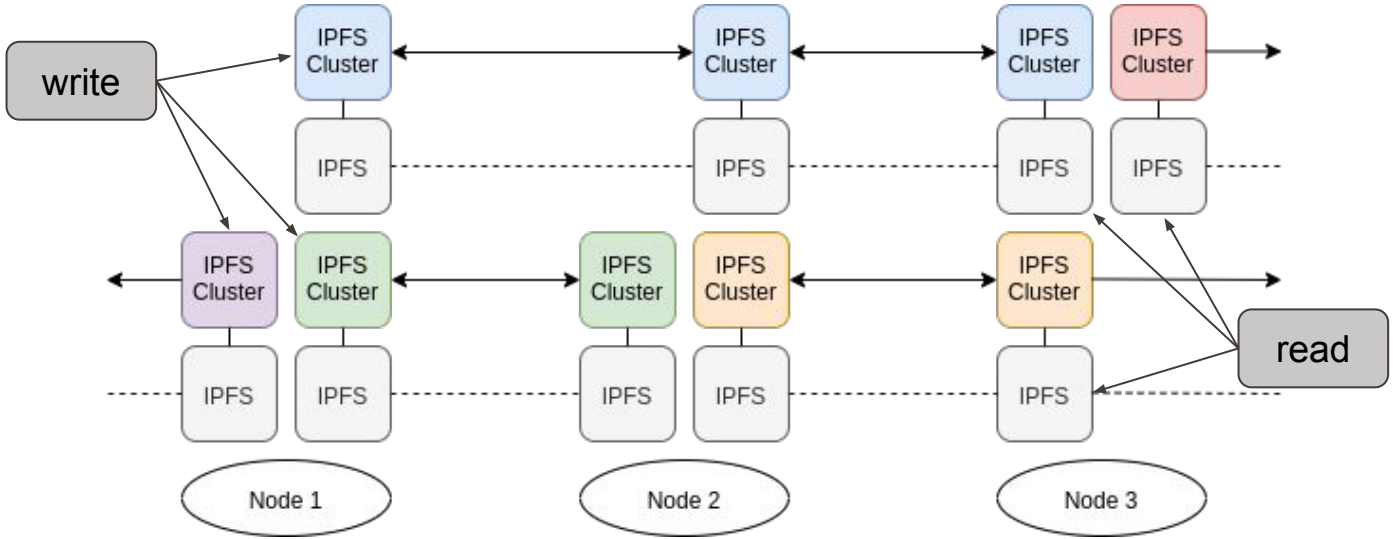
- Cruxified IPFS is faster than Vanilla IPFS for low latency pairs
- Improvement rate up to 18% on our experiment for Write+Read pair latency with RTT between nodes below 50 ms
- Future work on Cruxified IPFS:
  - Test partition resistance
  - Test responsiveness on a larger network

Design



Design


Node 2
Node 1
Node 3


write
IPFS Cluster
IPFS
IPFS Cluster
IPFS
IPFS Cluster
IPFS
IPFS Cluster
IPFS
IPFS Cluster
IPFS Cluster
IPFS
IPFS
IPFS Cluster
IPFS Cluster
IPFS
IPFS
IPFS Cluster
IPFS
read
Node 1
Node 2
Node 3

IPFS Identities

- Hash of public key
- PKI generation involving proof-of-work crypto puzzle making Sybils generation expensive, as detailed by S/Kademlia

  QmfDWkL9K1bEutSYd8wod8se7Z8AQnUav7EU1UKabBbYhk
- Can also be reached by its Multiaddress

  /ip4/104.236.76.40/tcp/4001

IPFS Names

- IPFS objects are:
  - Content addressed
  - Immutable

- Inter-Planetary Naming System (IPNS):
  - Register a name (initial hash of the content) when publishing an object
  - Pointer to the last version of the object

- DNSLink
  - Register a human readable name when publishing an object
  - DNS TXT record pointing to the last version of the object

“All problems in computer science can be solved by another level of indirection”

David Wheeler

Conflict-free Replicated Data Types (CRDTs)

- Strong Eventual Consistent (SEC)
  - Eventually consistent
  - Replicas that have delivered the same updates have equivalent state.
- Not sequentially consistent

CRDT Examples

Distributed counter with 2 operations: **ADD(N)** and **SUBSTITUTE(N)**

Initial state: 0

Node0:	*Node1*:
- **ADD(3)**	- **ADD(4)**
- **SUBSTITUTE(2)**	- **ADD(3)**
- **ADD(2)**	- **ADD(2)**
- **ADD(4)**	- **SUBSTITUTE(2)**
State: 3-2+2+4 = 7	State: 4+3+2-2 = 7

CRDT Examples

Distributed set with 2 operations: **ADD(e)** and **REMOVE(e)**

Initial state: { }

Node0:	*Node1*:
- ADD(e)	**- REMOVE(e)**
- REMOVE(e)	**- ADD(e)**
State: { e }	State: { e }

IPFS Cluster: CRDT

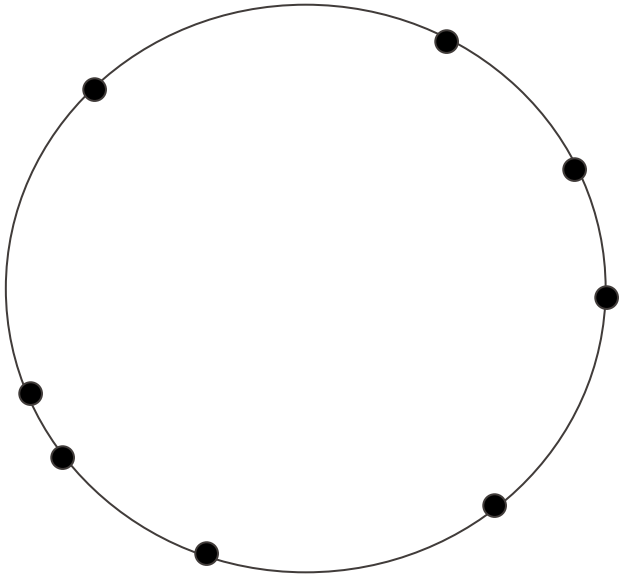
- Achieves Strong Eventual Consistency (SEC)
- Based on Merkle-CRDTs developed for the purpose of IPFS
- Using *Merkle-Clocks*, DAG logical clocks

IPFS Cluster: Raft

1. Leader majority election
2. Performing an update
   a. Send update to leader
   b. Leader broadcast update to all peers
   c. Peers acknowledge the update to the leader
   d. Once the leader has a majority of acks, the leader broadcast the confirmed update

Kademlia DHT

- XOR distance, clockwise circle
- k-buckets
- α concurrency parameter



Coral: Distributed Sloppy Hash Table (DSHT)

- Store data provider address in the DSHT
- Each key may have multiple values
- Coral proposes a hierarchical DSHT lookup

Key (content identifier)	Value (content provider)
<file1 hash>	124.12.53.212:9821
<file2 hash>	89.2.196.45:10412 133.251.66.149:6733
<file3 hash>	78.185.4.22:7822

Available Responsive Areas (ARAs)