Cross-DC Fault-Tolerant ViewFileSystem @Twitter

Gera Shegalov

@gerashegalov
Staff Software Engineer
PIE: Product Insight & Experimentation
Committer @ApacheHadoop

Ming Ma

@mingmasplace
Staff Software Engineer
@TwitterHadoop
Committer @ApacheHadoop

Outline

- Multi-cluster usage
  - Setup of hadoop clusters
  - Usage patterns
- Cross-DC ViewFileSystem design and implementation
  - #dcp: Many clusters, one namespace
  - #nfly: Transparent read/write from/to multiple cluster paths
- Next steps

Hadoop workload @Twitter

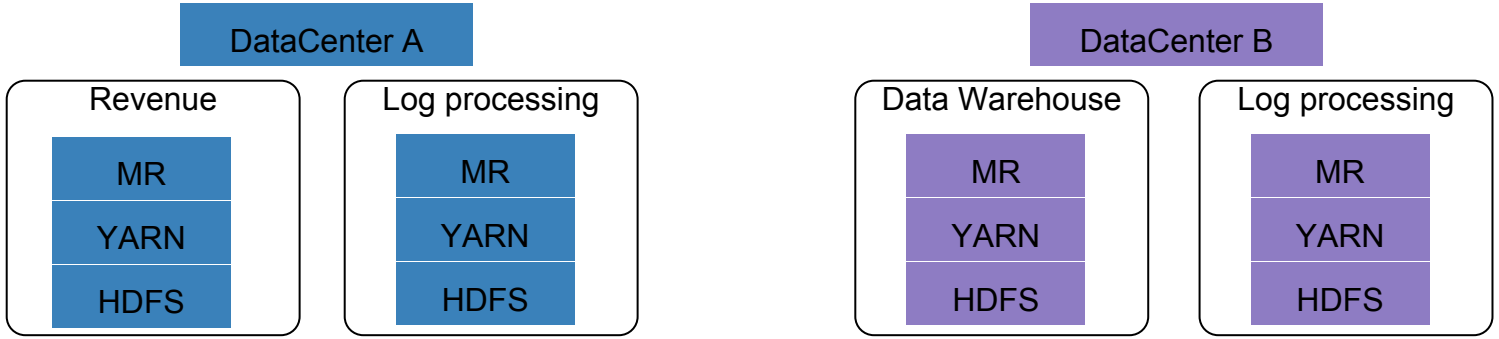
- Large scale
  - Thousands of machines
  - 150K+ jobs / day
- Diverse workload
  - Production vs ad-hoc
  - Batch vs interactive vs iterative
- Cross Data Centers
- Require scalability, performance isolation

One giant cluster?

- Can’t run across DCs
- Scalability issue with hadoop master daemons
- Performance isolation
- Support for different major versions
- .....

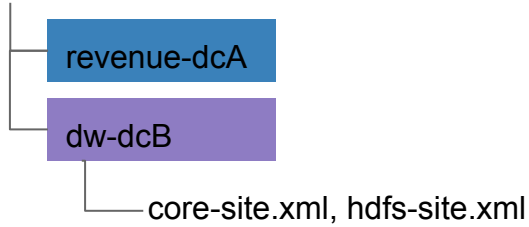
Use of multiple clusters

- Vertical partitioning



- Usage and configuration management

Directory layout on local file system
/etc/hadoop



Command Line Interface

```
hadoop —config /etc/hadoop/revenue-dcA fs -ls /
hadoop —config /etc/hadoop/dw-dcB fs -ls /
……
```

Use of multiple clusters - continued

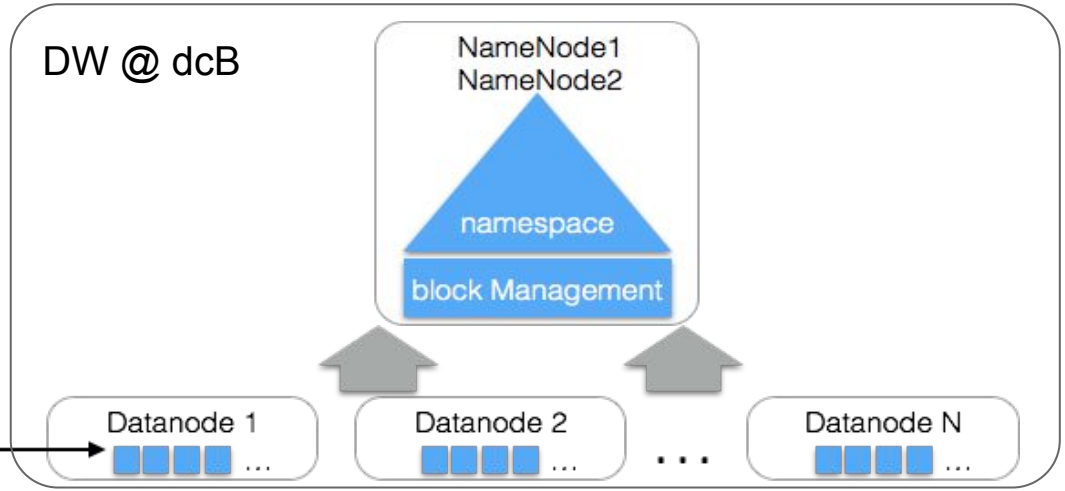
- Benefits
  - Scalability
  - Cross-DC support
  - Good isolation

Use of multiple clusters - continued

- Benefits
  - Scalability
  - Cross-DC support
  - Good isolation
- Issues
  - Harder to use, operate and support
  - Efficiency due to lack of elasticity and over replication

HDFS Without Federation

Client machine: /etc/hadoop/dw-dcB/hdfs-site.xml, core-site.xml



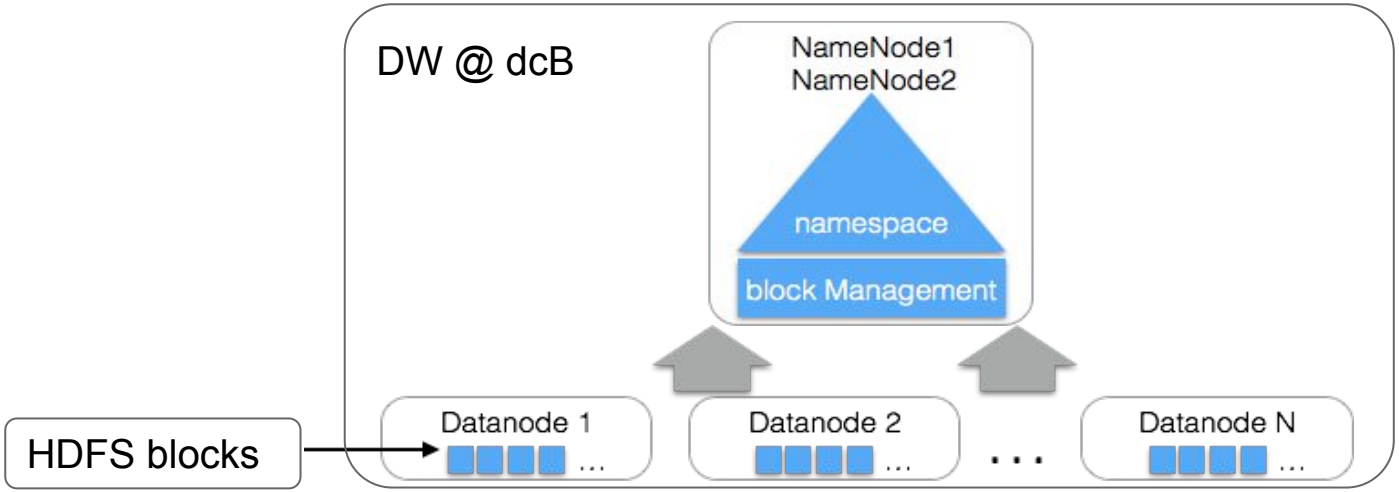
HDFS Without Federation

Client machine: /etc/hadoop/dw-dcB/hdfs-site.xml, core-site.xml

hadoop -config /etc/hadoop/dw-dcB fs -get /user/bob/fileB

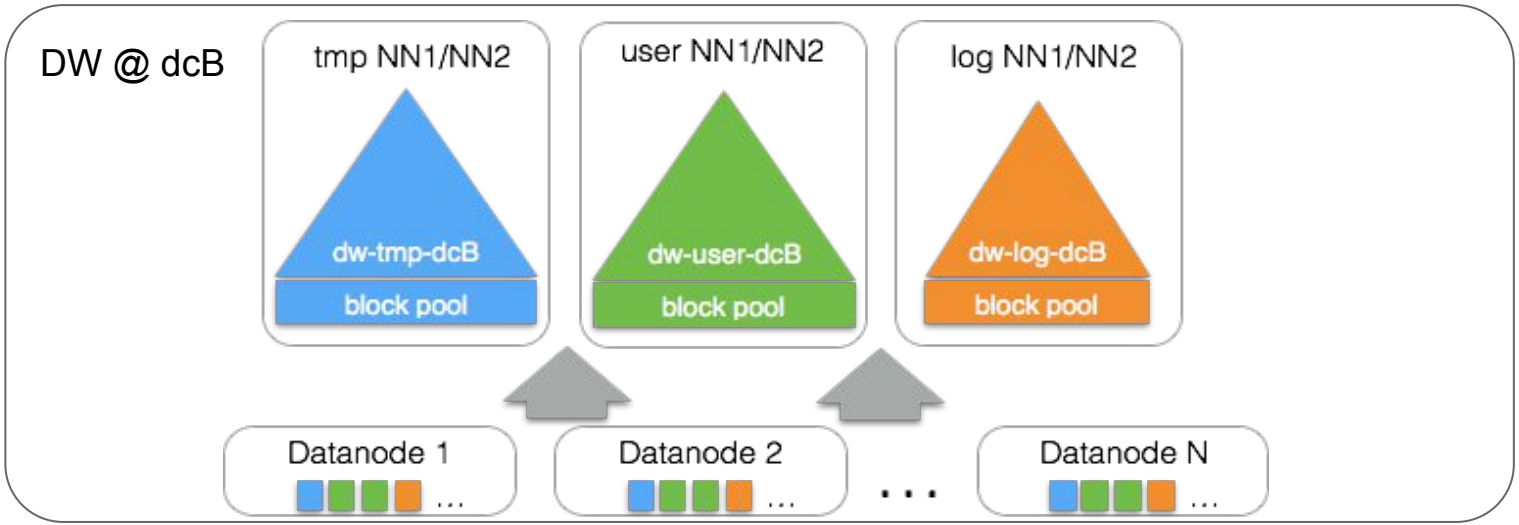
Distributed FileSystem

fs.defaultFS -> hdfs://dw-nn-dcB -> NameNode1, NameNode2



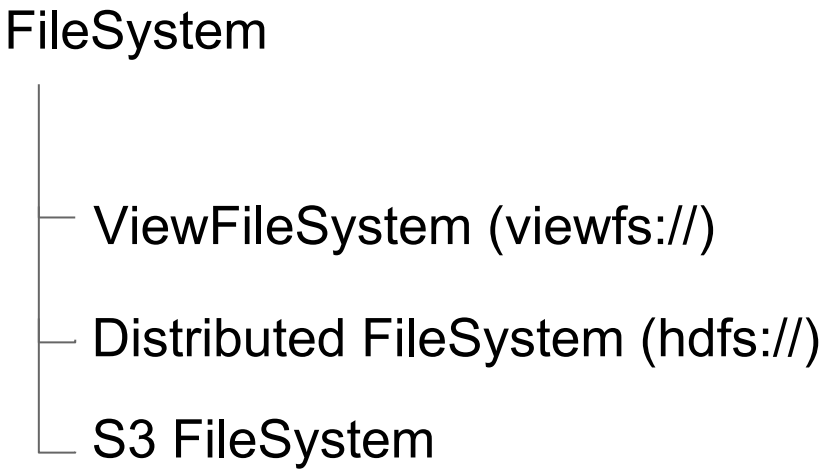
HDFS Federation: partitioning

- Global namespace partitioned into disjointed namespaces
- One block pool per namespace
- One or more block pools on each datanode



HDFS Federation: abstraction

- Transparent global view via client-side mount table
- Implemented as FileSystem-based ViewFileSystem



mountable.xml

App paths	HDFS paths
viewfs://dw-nn-dcB/var	hdfs://dw-tmp-dcB/var
viewfs://dw-nn-dcB/user	hdfs://dw-user-dcB/user
viewfs://dw-nn-dcB/logs	hdfs://dw-log-dcB/logs

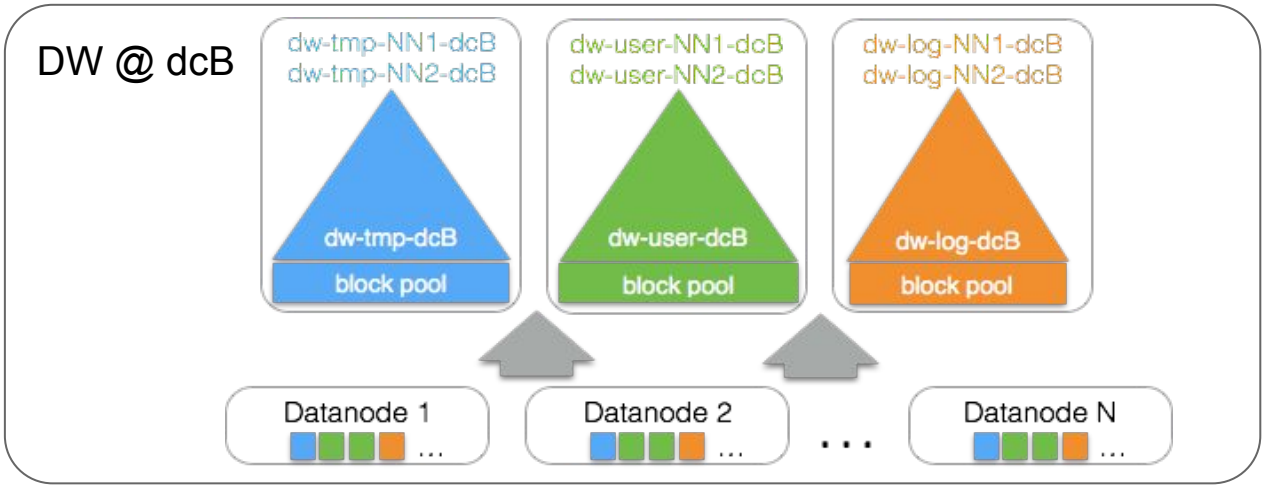
HDFS Federation Example

Client machine: /etc/hadoop/dw-dcB/hdfs-site.xml, core-site.xml, mountable.xml

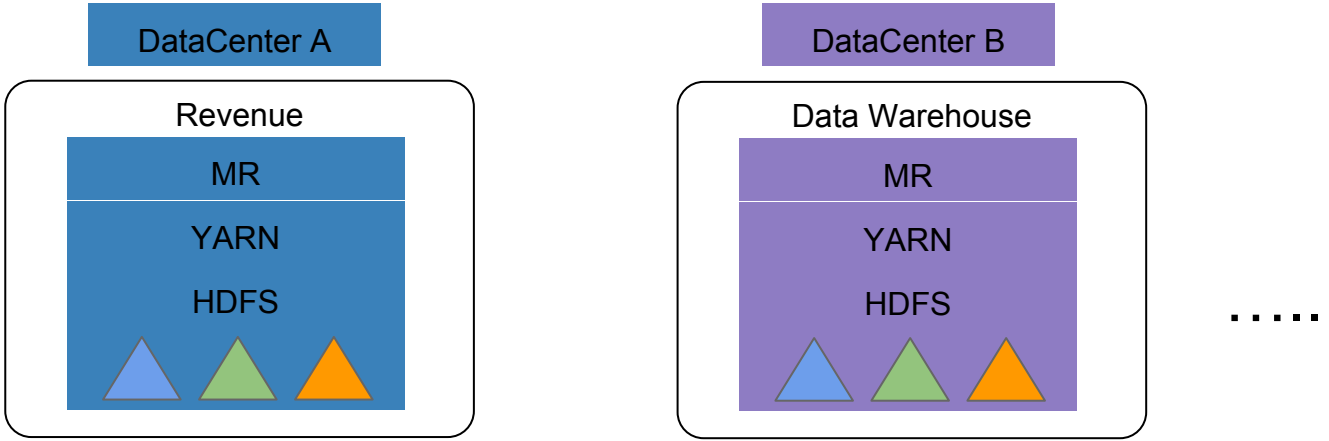
hadoop -config /etc/hadoop/dw-dcB fs -get /user/bob/fileB

ViewFileSystem
/user -> hdfs://dw-user-dcB/user

Distributed FileSystem
dw-user-dcB -> dw-user-NN1-dcB, dw-user-NN2-dcB



HDFS Federation + Multiple clusters



- Twitter scale:
  - Several namespaces / cluster
  - Thousands of nodes / cluster

Multi-cluster usage patterns

- Pattern: Per-cluster operation
- Examples:

```
hadoop —config /etc/hadoop/revenue-dcA fs -get /user/bob/fileA
```

Multi-cluster usage patterns

- Pattern: Per-cluster operation
- Examples:

```
hadoop —config /etc/hadoop/revenue-dcA fs -get /user/bob/fileA

hdfs --config /etc/hadoop/dw-dcB fsck -fs hdfs://dw-user-dc2 /
```

Multi-cluster usage patterns

- Pattern: Per-cluster operation
- Examples:

```
hadoop —config /etc/hadoop/revenue-dcA fs -get /user/bob/fileA

hdfs --config /etc/hadoop/dw-dcB fsck -fs hdfs://dw-user-dc2 /

// find all “fileC” files on all clusters
for i in `ls /etc/hadoop`;
do hadoop --config /etc/hadoop/$i fs -ls fileC; done
```

- Issue: Usability

Multi-cluster usage patterns

- Pattern: Cross-cluster asynchronous data replication
- DistCp & HDFS Federation
    - Example: copy data from revenue cluster to dw cluster via “hadoop -config /etc/hadoop/dw-dcB distcp **$sourcePath** /logs/dirB”

Multi-cluster usage patterns

- Pattern: Cross-cluster asynchronous data replication
- DistCp & HDFS Federation
  - Example: copy data from revenue cluster to dw cluster via “hadoop -config /etc/hadoop/dw-dcB distcp **$sourcePath** /logs/dirB”

Description	$sourcePath	Result
global namespace	viewfs://revenue-nn-dcA/logs/dirB /logs/dirB	Source path unresolvable

Multi-cluster usage patterns

- Pattern: Cross-cluster asynchronous data replication
- DistCp & HDFS Federation
  - Example: copy data from revenue cluster to dw cluster via “hadoop -config /etc/hadoop/dw-dcB distcp **$sourcePath** /logs/dirB”

Description	$sourcePath	Result
global namespace	viewfs://revenue-nn-dcA/logs/dirB /logs/dirB	Source path unresolvable
Active namenode	hdfs://revenue-log-nn1-dcA/logs/dirB /logs/dirB	not reliable due to hard coded namenode

Multi-cluster usage patterns

- Pattern: Cross-cluster asynchronous data replication
- DistCp & HDFS Federation
  - Example: copy data from revenue cluster to dw cluster via “hadoop -config /etc/hadoop/dw-dcB distcp **$sourcePath** /logs/dirB”

Description	$sourcePath	Result
global namespace	viewfs://revenue-nn-dcA/logs/dirB /logs/dirB	Source path unresolvable
Active namenode	hdfs://revenue-log-nn1-dcA/logs/dirB /logs/dirB	not reliable due to hard coded namenode
hftp & DNS alias	hftp://revenue-nn-dcA/logs/dirB	hftp reliability and efficiency

- Issues: Usability, reliability and efficiency

Multi-cluster usage patterns

- Pattern: Cross-DC strong consistency
- Scenario:
  - Data written to two or more DCs synchronously
  - Consistency & Error handling
  - Read & Data locality
- Issue: Functionality not available in hadoop

PHASE 1: Hackweek project #DCP

Initial state

- Incompatible 1.x and 2.x versions
- Mixture of **hdfs**, **viewfs**, **hftp** URI's, **s3** data on AWS
- Hftp is read-only, unaware of HDFS HA, and Federation
- Hftp used even for copying between compatible clusters

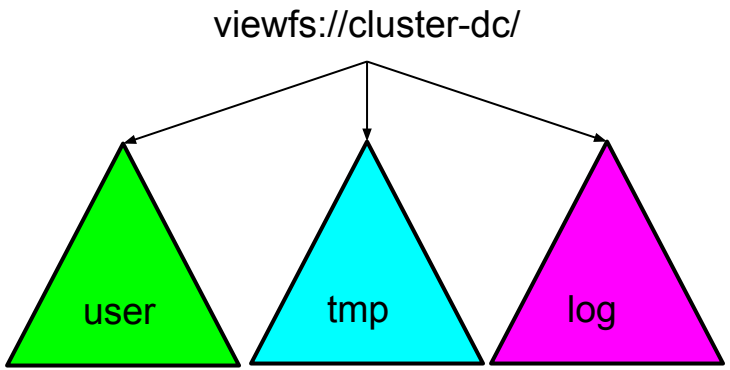
Very frequently asked support questions

- Why does my job fail with a Read-Only exception?
- We can ‘fs -ls’ input but the job fails, why?
- Can’t connect to hdfs/viewfs://cluster-nn:**50070**/
- When used with DistCp is **hftp** for source or destination?

How can we abstract all the different URI's?

1.x clusters had a single namespace

2.x clusters typically have at least **three** namespaces for /user, /tmp and /logs and we already used viewfs to mask their existence.



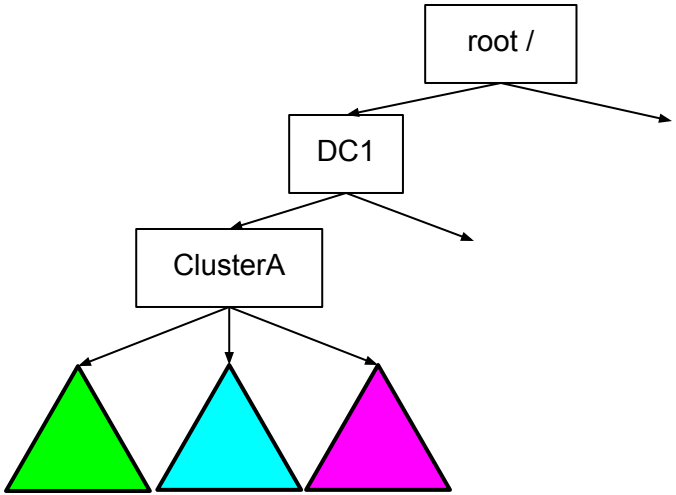
One namespace for all of Twitter filesystems

2.x clusters are already more of a deployment unit

- 1 YARN cluster but
- 3 HDFS namespaces (clusters) behind viewfs://**dc-cluster**/

Why not have all HDFS clusters in this view?

Why not have the node-local home and tmp or s3 in this view?



How to create a global conf for the Twitter filesystems?

Single physical conf is impractical

- the opposite of **divide and conquer**

Factor out “exportable” conf keys and xinclude it in other confs

- Deployment at our scale is already complex and no easy way to do it
- Xinclude usage is rare and easily hits bugs, e.g., YARN-1741

Challenge: Sometimes easier to change software than config management

#dcp: DistCp with less typing

Opportunity: already have code **TwitterViewFs** to handle two schemes

- viewfs://cluster-dc/path
- hdfs://cluster-dc/path

TwitterViewFs is initialized first as **fs.defaultFS**

All configs are available at the client gateway nodes:

- List glob **file:/etc/hadoop/hadoop-conf-*** or any other glob
- generate global conf on the fly
- FileSystem#getConf as the base conf for Scalding REPL and jobs
- No changes for the cluster side config, thanks to **job.xml**

What's involved in generating the global config?

- **core-site.xml**: ViewFs mount tables to construct the global namespace
  - Preserving the **original paths** of the cluster: **/user/cecily**
  - Add **/dc_i/cluster_j/user/cecily** for all clusters **j** in DC **i**

- Replace target 1.x cluster hdfs:// URI's with **hftp://**

- Include node's local /home/$USER and /tmp/hadoop-$USER under /local

- **hdfs-site.xml**: Merging 2.x DFSClient configs (HA nameservices, etc)

Constructing fs.viewfs.mounttable namespace (1)

For the original link: **c1-dc2**.link./user -> hdfs://userURI_c1dc2

if (c1-dc2 is Hadoop 2.x) then add:

c1-dc2.link./**dc2**/**c1**/user -> hdfs://userURI_c1dc2

else // if (c1-dc2 is Hadoop 1.x) then

c1-dc2.link./**dc2**/**c1**/user -> **hftp**://userURI_c1dc2

```
scalding> fsShell("-ls /dc*/")        // list all clusters
Found N items
...
```

Constructing fs.viewfs.mounttable namespace (2)

Add local mounts for the current user:

c1-dc2.link./**local**/user/${user.name} -> file:///home/${user.name}

c1-dc2.link./**local**/tmp/hadoop-${user.name} -> file://${hadoop.tmp.dir}

```
// no need to remember copy(From/To)Local aka get/put
scalding> fsShell("-cp /local/user/cecily/file.txt
                      /user/cecily/hdfs_file.txt")
    res6: Int = 0
```

Merging DFSClient configurations

dfs.nameservices: build a comma-separated list from all hdfs-site.xml

hdfs-site.xml supports multi-namespace config via suffixes

for each **dfs.nameservices** copy keys containing the nameservice id, such as:

- dfs.client.failover.proxy.provider.nameservice
- dfs.ha.automatic-failover.enabled.nameservice
- dfs.namenode.rpc-address.nameservice.{nn1,nn2}

Putting it all together

show user's quota usage across all clusters

```
$ hadoop fs -count -q '/dc*/user/cecily'
```

run fsck without thinking about individual namespace URI's on c1-dc1

```
$ hadoop fsck /dc1/c1/user/cecily
```

Most DistCp is from a production cluster to the default ad-hoc cluster

```
$ hadoop distcp /dc1/prod/user/cecily/model1 /user/cecily/model1
```

Pull data in REPL without extra copying to the default cluster

```
scalding> MySource("/dc1/prod/logs/some_events").toIterator.filter{...}
```

User testimonials

“This is totally awesome!”

“Everytime I use the cross-DC dcp, I appreciate the work gera did with go/dcp. It's just soooooooo good”

“#dcp that says it all”

PHASE 2: #Nfly aka HADOOP-12077

Dictionary:

Nfly is something about N x DC-FileSystem

Nfly paths for cluster outages which never happens :-)

Micro-services store ML models, config, etc in a cluster role cl in *n* DC's:

cl-dc1 and **cl-dc2**

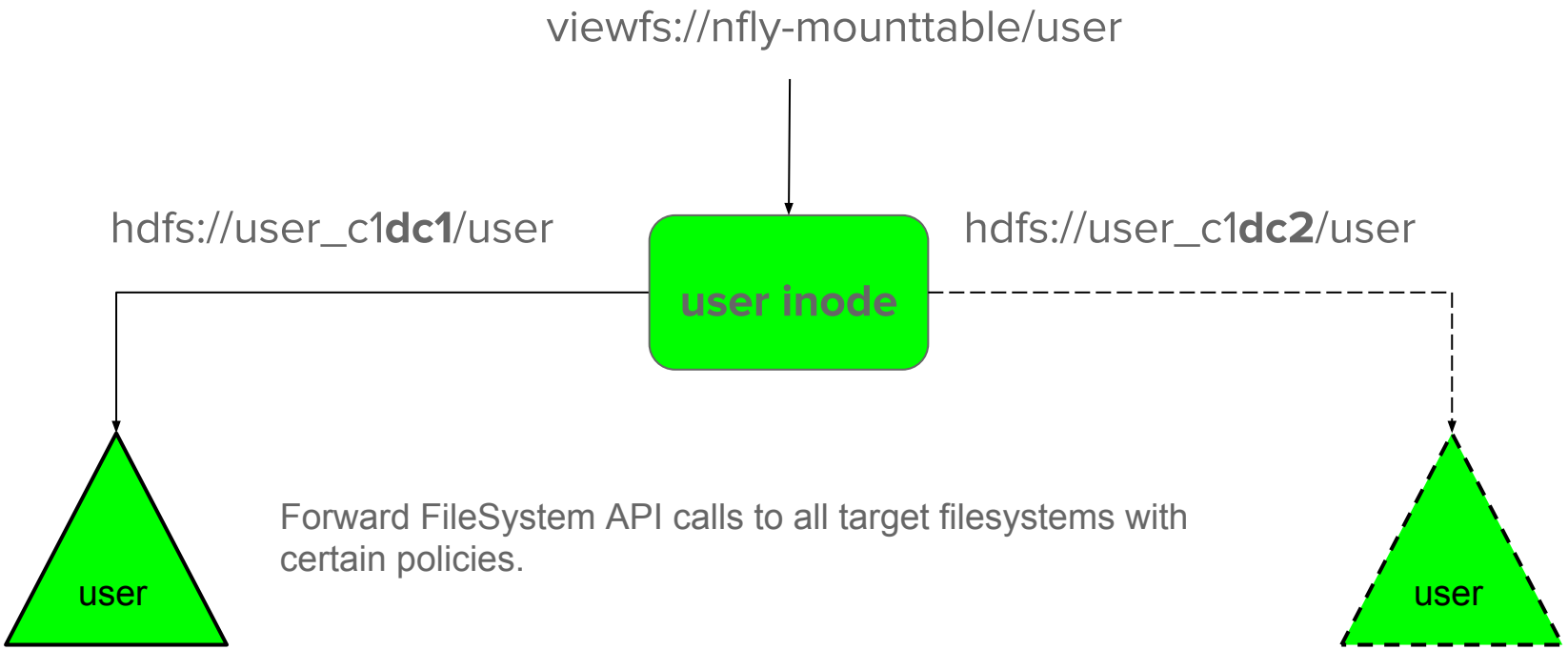
Avoid having service owners write failover logic. Common read patterns:

- Read from the nearest HDFS (e.g., same DC)
- Read most recent "replica" from the nearest HDFS

So far in ViewFs:

1 virtual inode/path == **1** "physical" URI

What if we had a multi-URI inode



Read policies: nearest cluster

- Client services are typically on Mesos, no overlap
- Use hadoop **NetworkTopology** to sort the client node and target URI's authorities, same way as HDFS client does with
- Challenge: targets are HA nameservices, not hostnames
- Solution: nameservice name incl. the DC name, topology script can resolve it to **/dc/nameservice**
- DC-local "OFF-SWITCH" locality is all we care about

Read policies: most recent, nearest cluster

Given a read-only API call for *path*, query getFileStatus for *path*

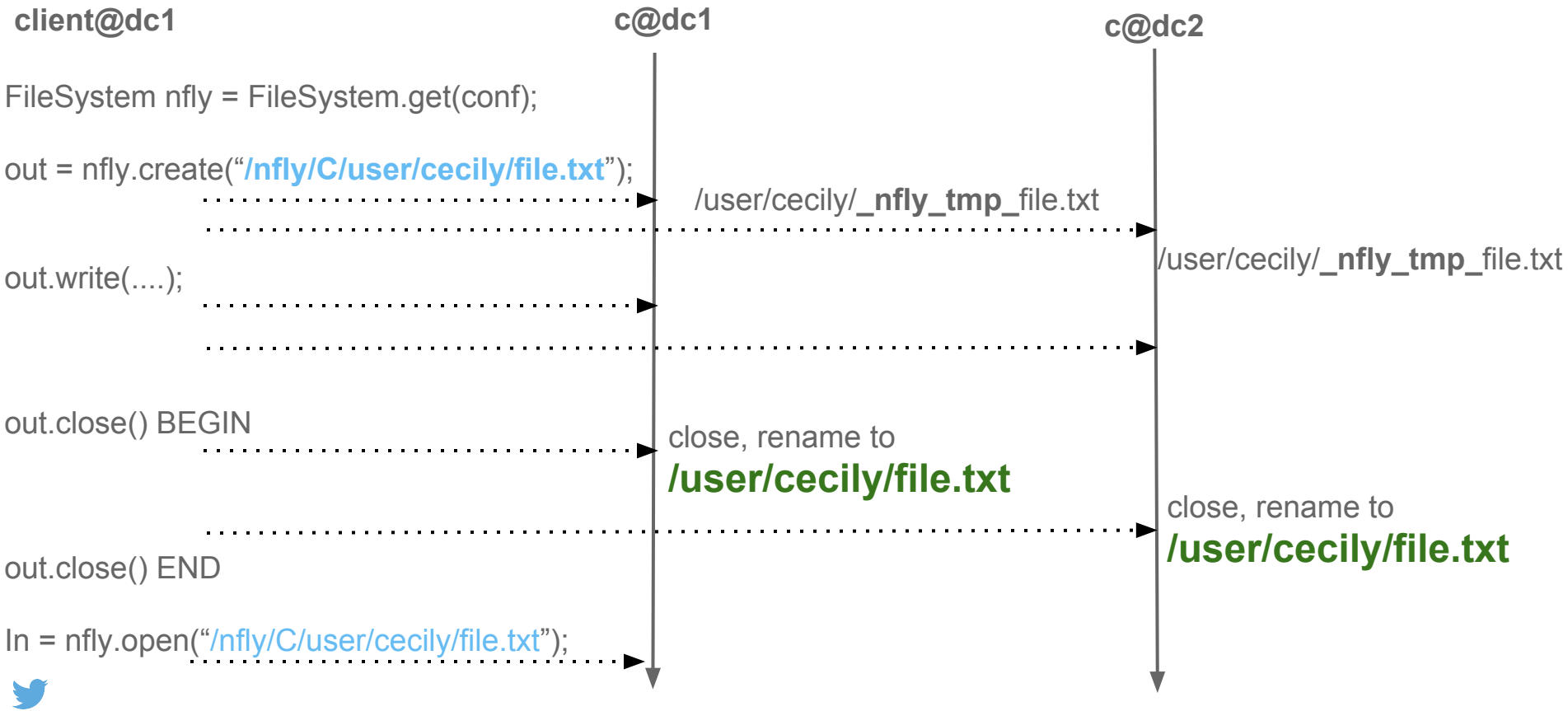
- Sort by mtime,
- secondary sort by *NetworkTopology*

Much more expensive due to RPC to all URI

Additional option: **read-repair**

- Copy the best replica to stale or missing destinations

Write policy: synchronous with min replication



Challenges of the client-driven replication

- No pipelining, node's NIC is the bottleneck
  - OK because this is not about big data use case
- Client dies: Who picks up the trash?
  - Dedicated retention management policies expire files on our clusters
- Min replication reached, how are the remaining ones recovered?
  - Read-repair
  - Can set up internal tool to synchronize replicated destinations

Future work

Our work drew attention from the community, and more people implemented our approach, especially the namespace component.

More work is underways in OSS, e.g.:

HDFS-10467: Router-based HDFS federation

Summary

- Multi-DC use cases at Twitter
- #dcp Improved usability through the single namespace
- #nfly Pragmatic approach to replication via client-side multi-URI viewfs inodes across DC

Acknowledgements

- Current and former members of @TwitterHadoop, in particular:
- Laurent Goujon
- Lohit VijayaRenu

Thank you

questions.map { q => answer(q) }