Using XRootD to Minimize Hadoop Replication

Jeff Dost UCSD

Introduction

- The UCSD T2 uses the Hadoop Distributed File System (HDFS) for our Storage Element
- Hadoop cuts a file into small chunks known as blocks
- These blocks are distributed throughout the cluster
- Blocks are replicated N times to provide redundancy
- The UCSD T2 is also a part of the AAA CMS XRootD Federation
- The Federation provides global redundancy for files that are available at multiple CMS sites

HDFS XRootD Fallback

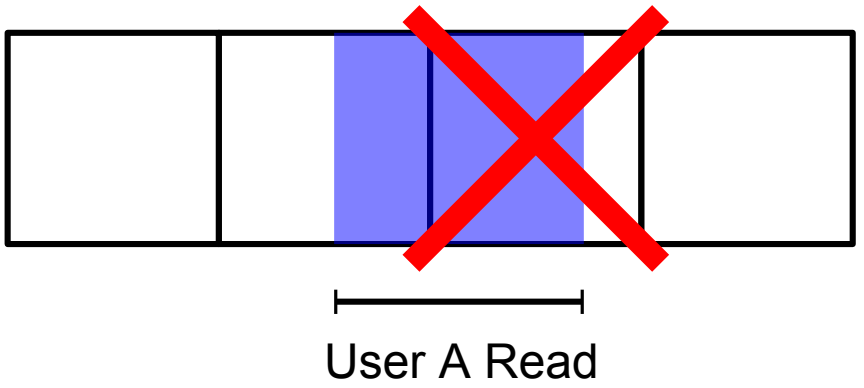
- Local Hadoop replication is expensive!
  - If N=2 we only get to utilize half our physical disk for storage
- Why not take advantage of the global replication the XRootD Federation already provides us?
- At UCSD we have developed a system that does exactly this, the **HDFS XRootD Fallback system**
- HDFS Fallback allows a site to reduce Hadoop replication to 1, freeing up considerable local physical space

HDFS XRootD Fallback

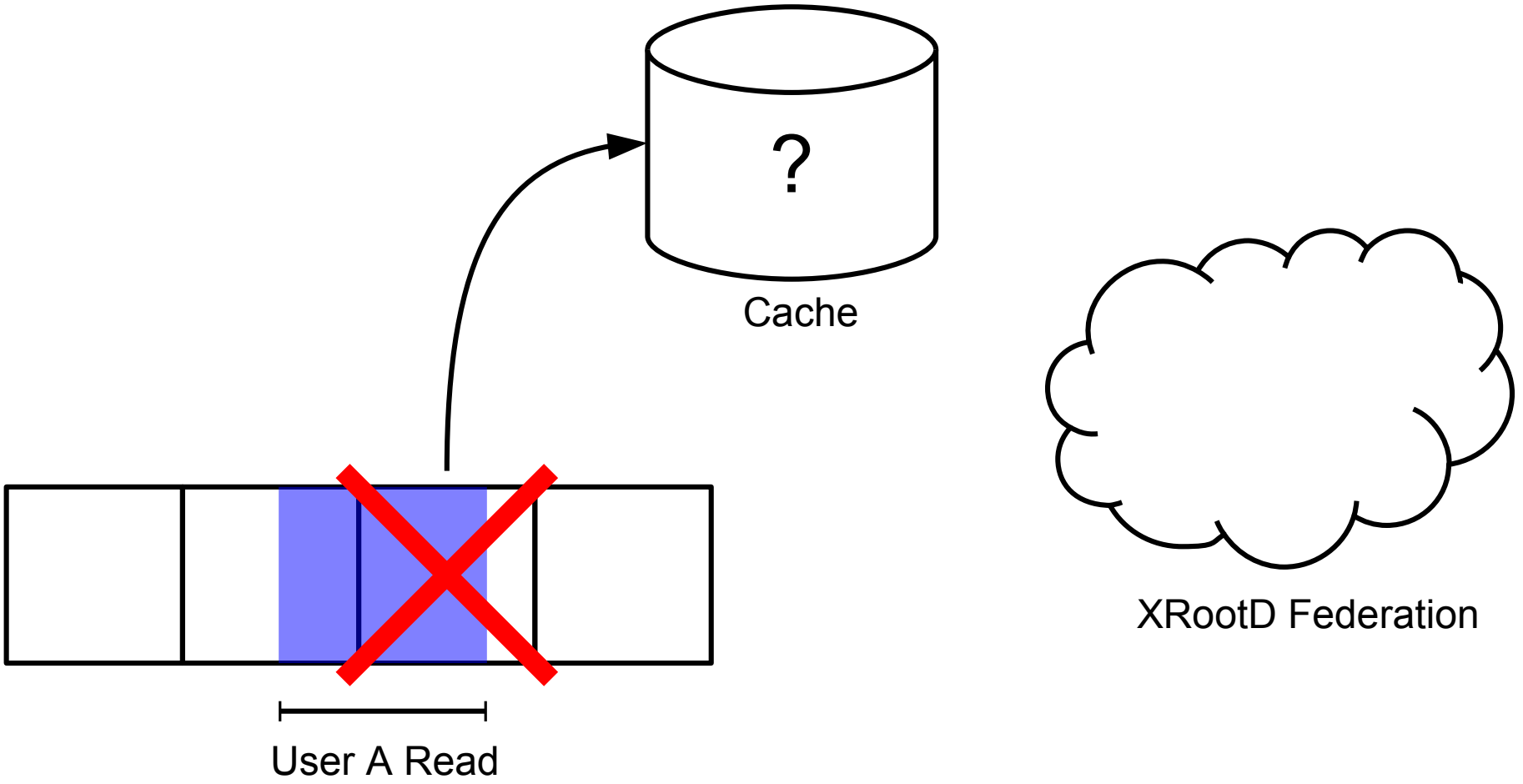
- The architecture is composed of two major components:
- **hdfs-xrootd-fallback** – accesses broken blocks on demand via XRootD, and caches them locally for future requests
- **hdfs-xrootd-healer** – injects fallback-fetched blocks back into hadoop to repair broken files in local storage

Fallback Architecture

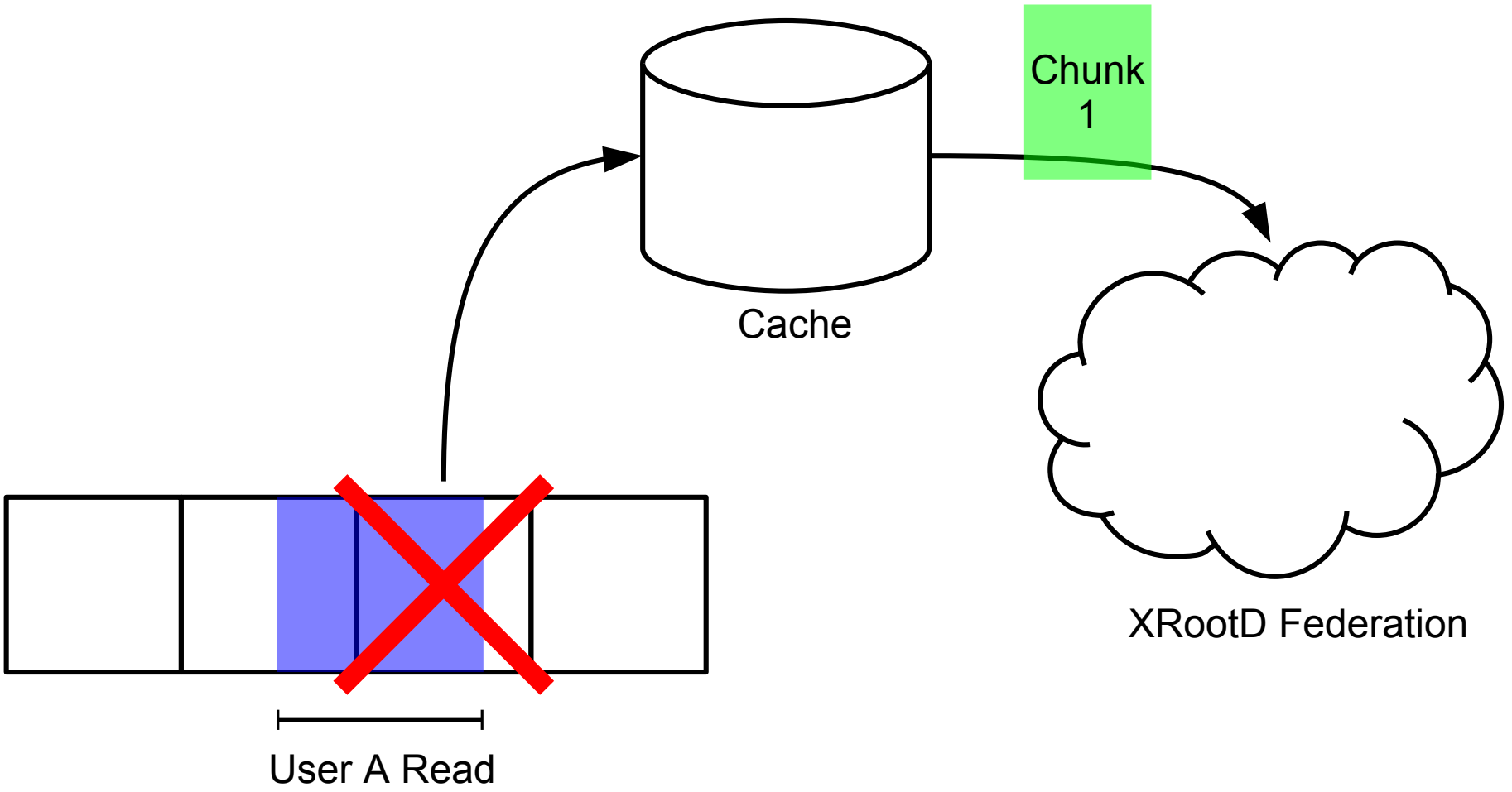
- In Hadoop, if a file block cannot be located anywhere, the read will throw an IOException when a user program tries to access that portion of the file



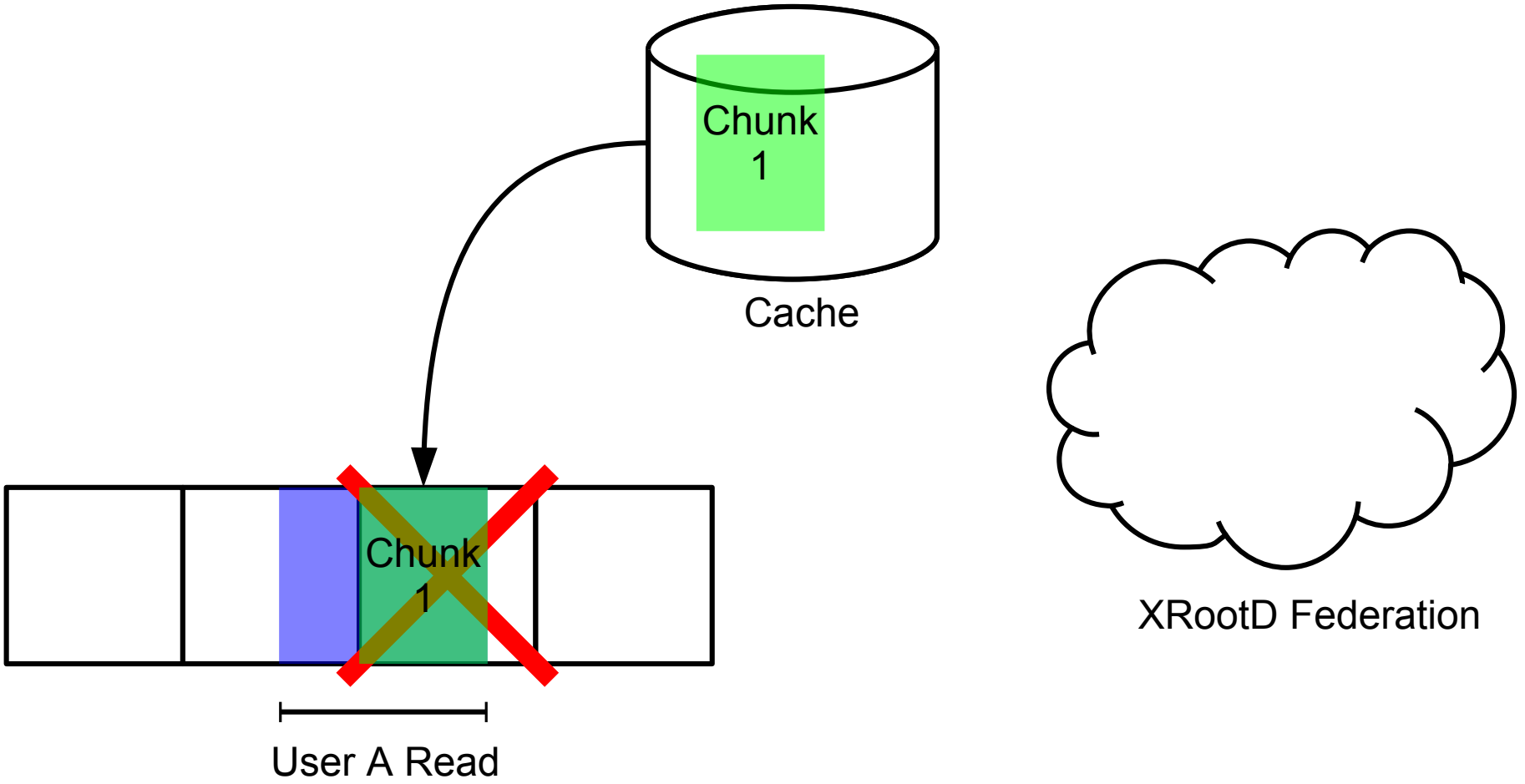
Fallback Architecture



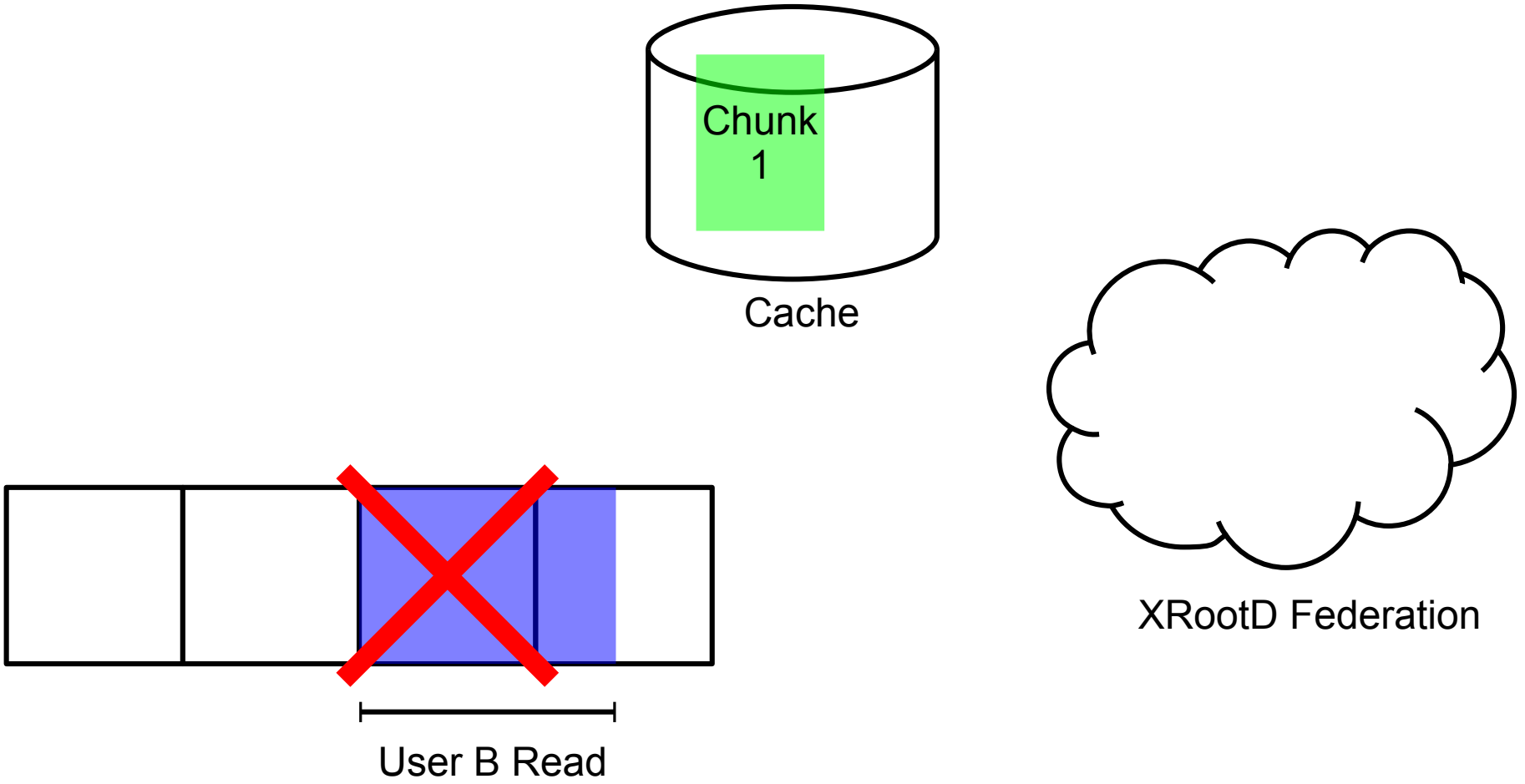
Fallback Architecture



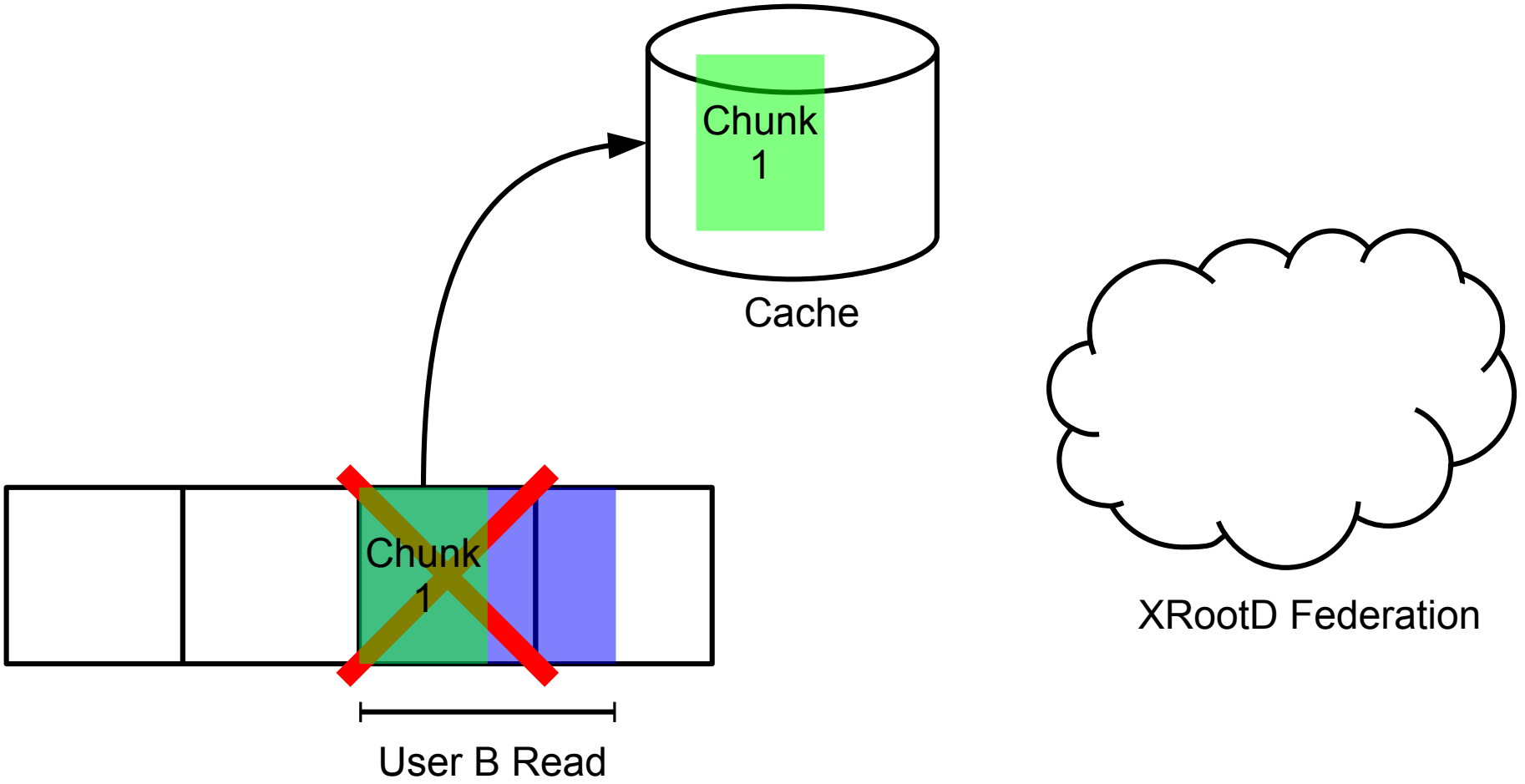
Fallback Architecture



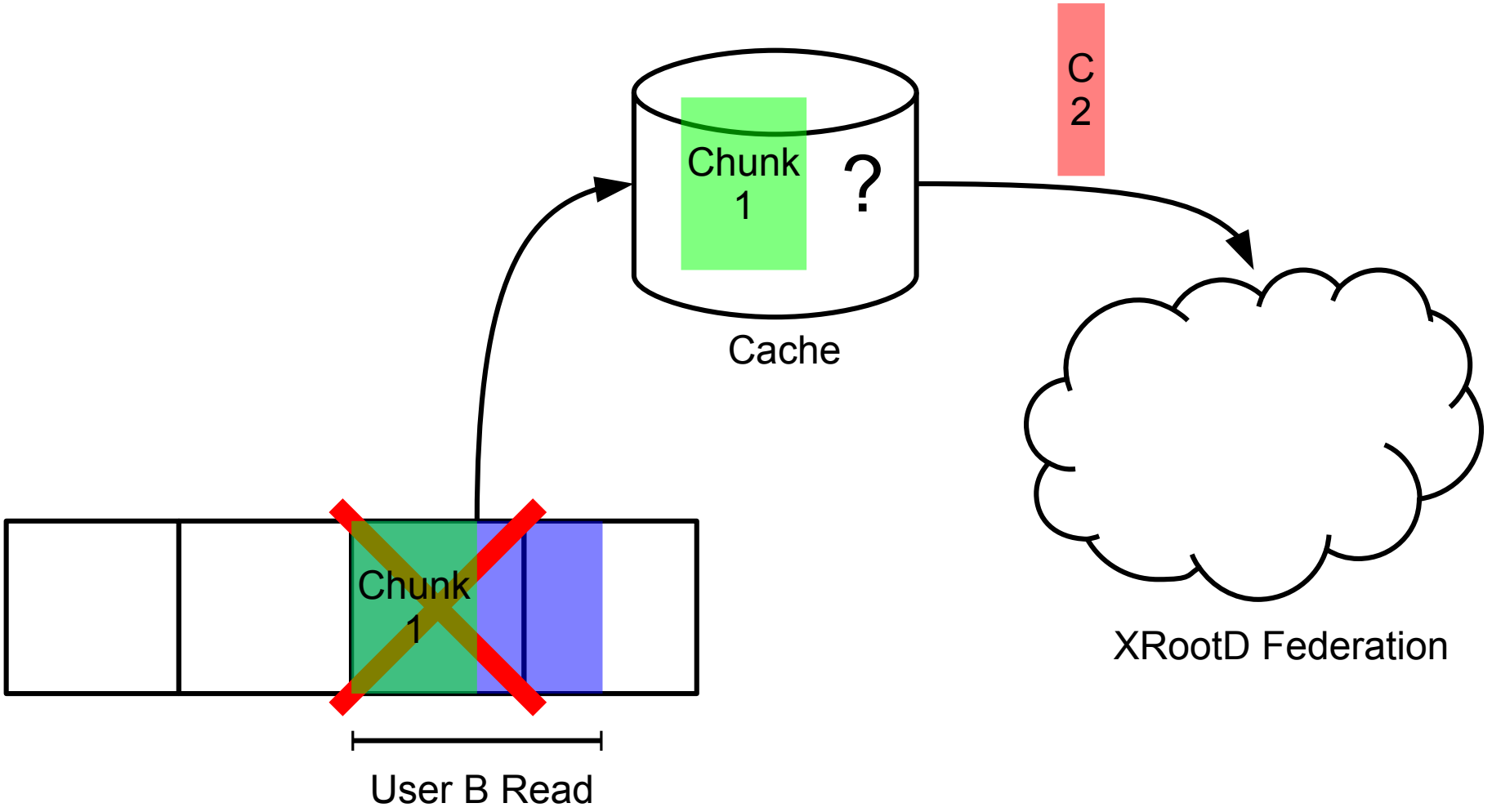
Fallback Architecture



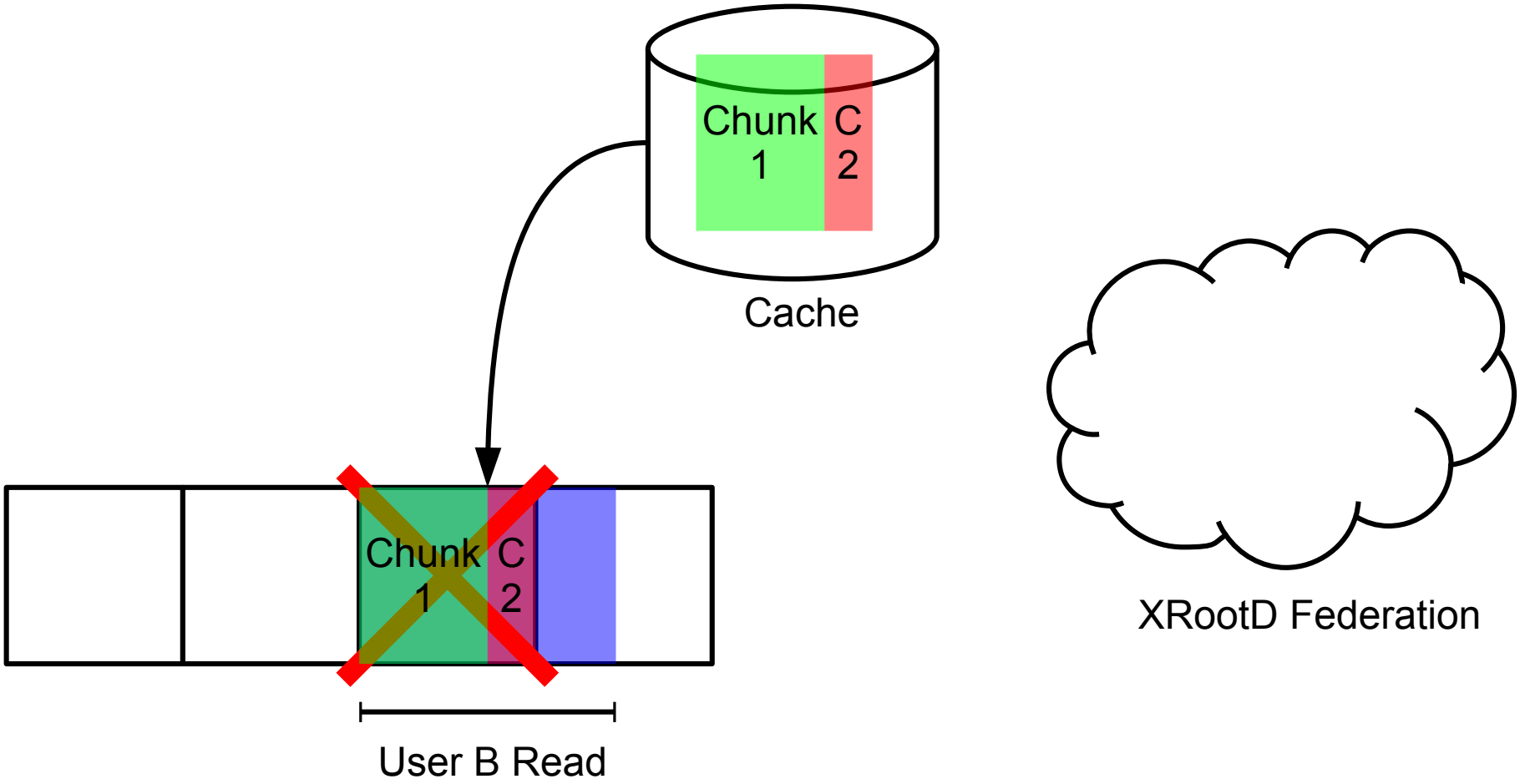
Fallback Architecture



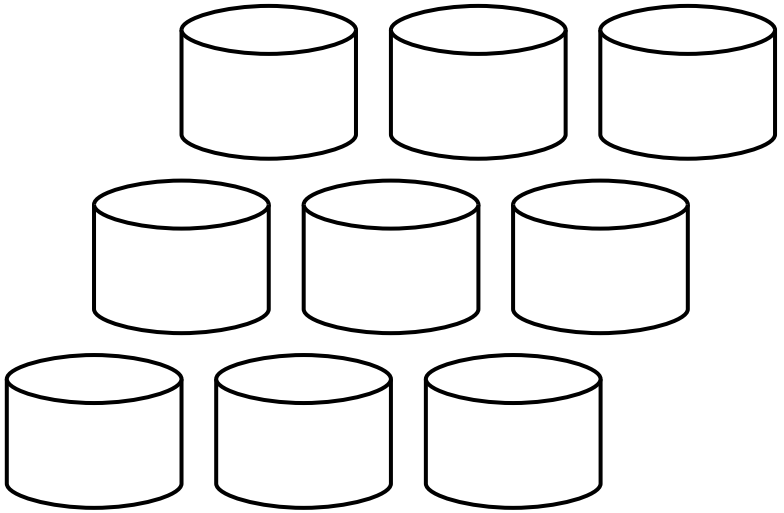
Fallback Architecture

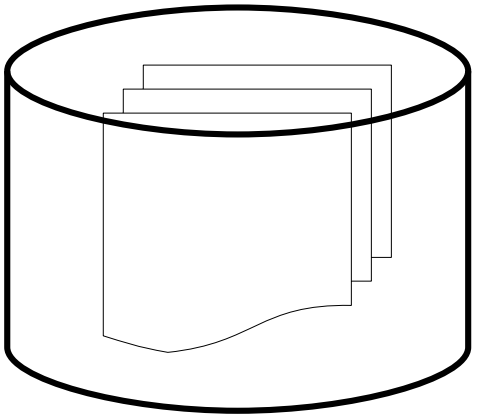


Fallback Architecture



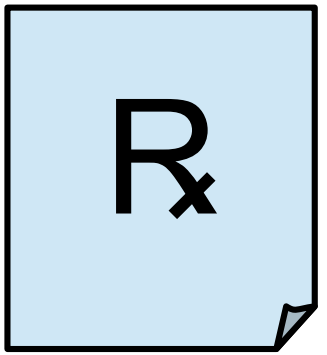
Healer Architecture



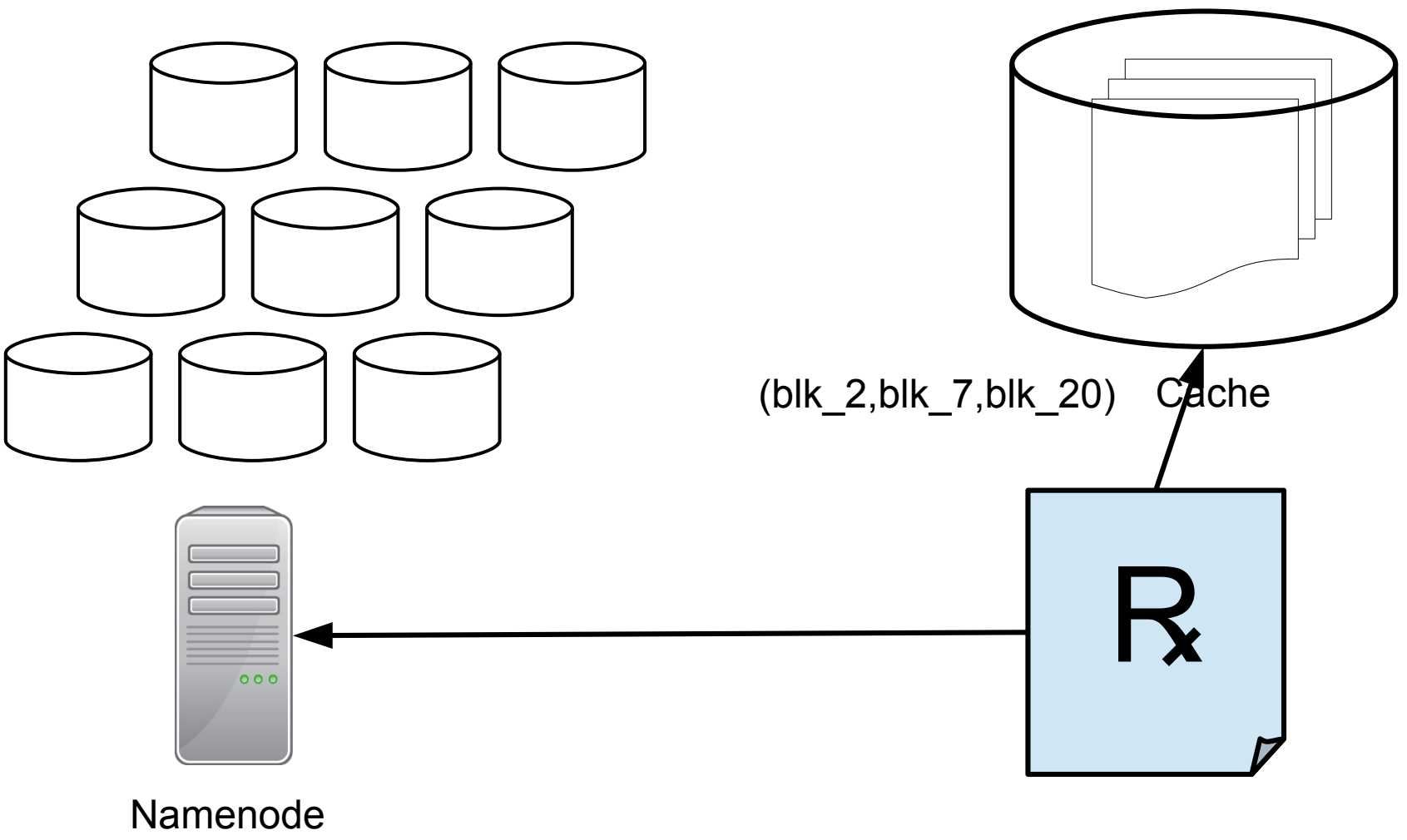


Cache

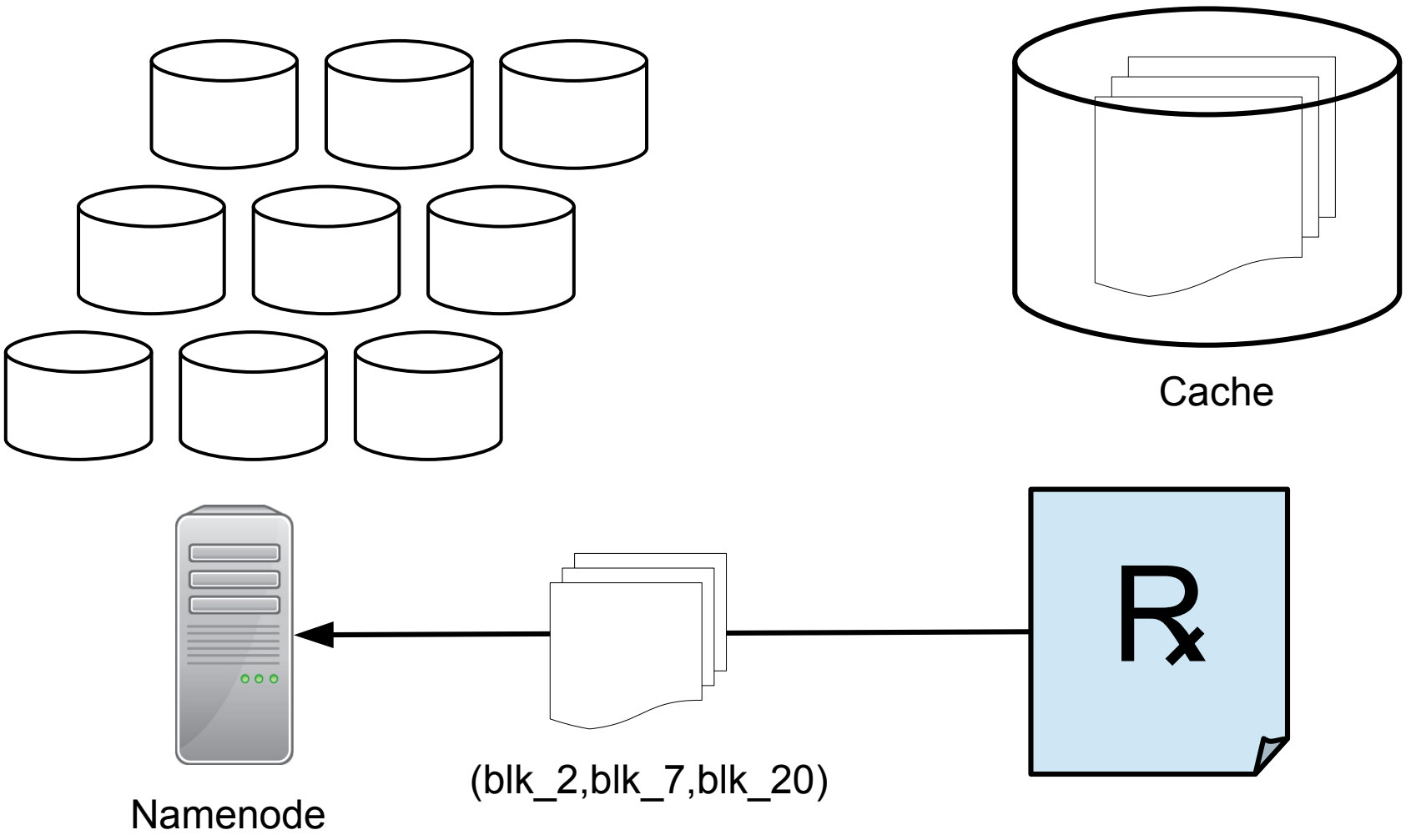
Namenode



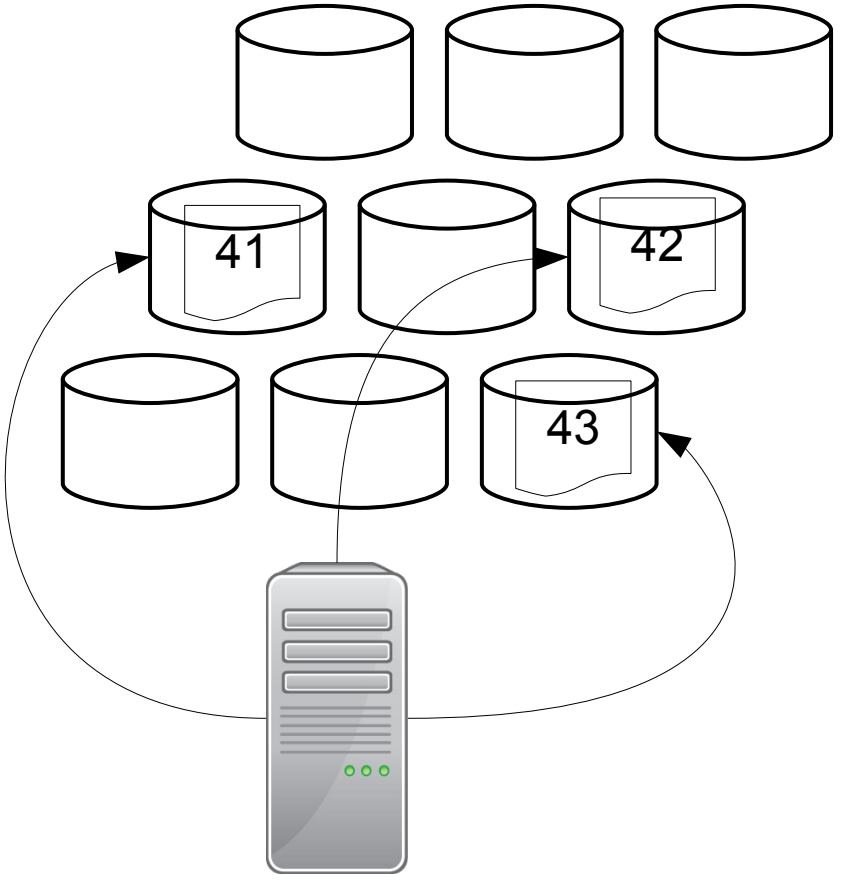
Healer Architecture

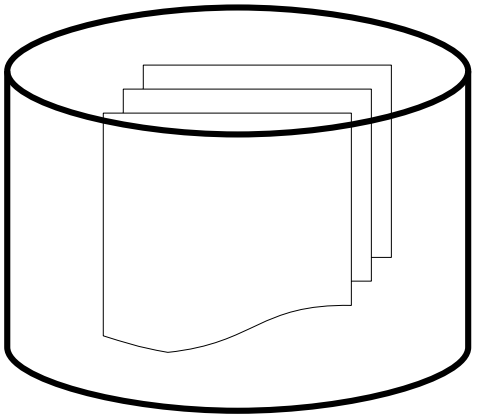


Healer Architecture

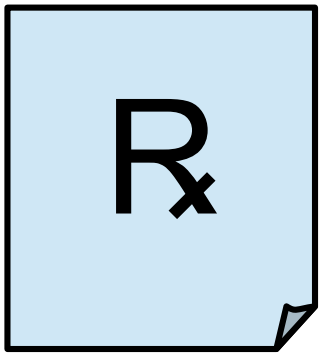


Healer Architecture

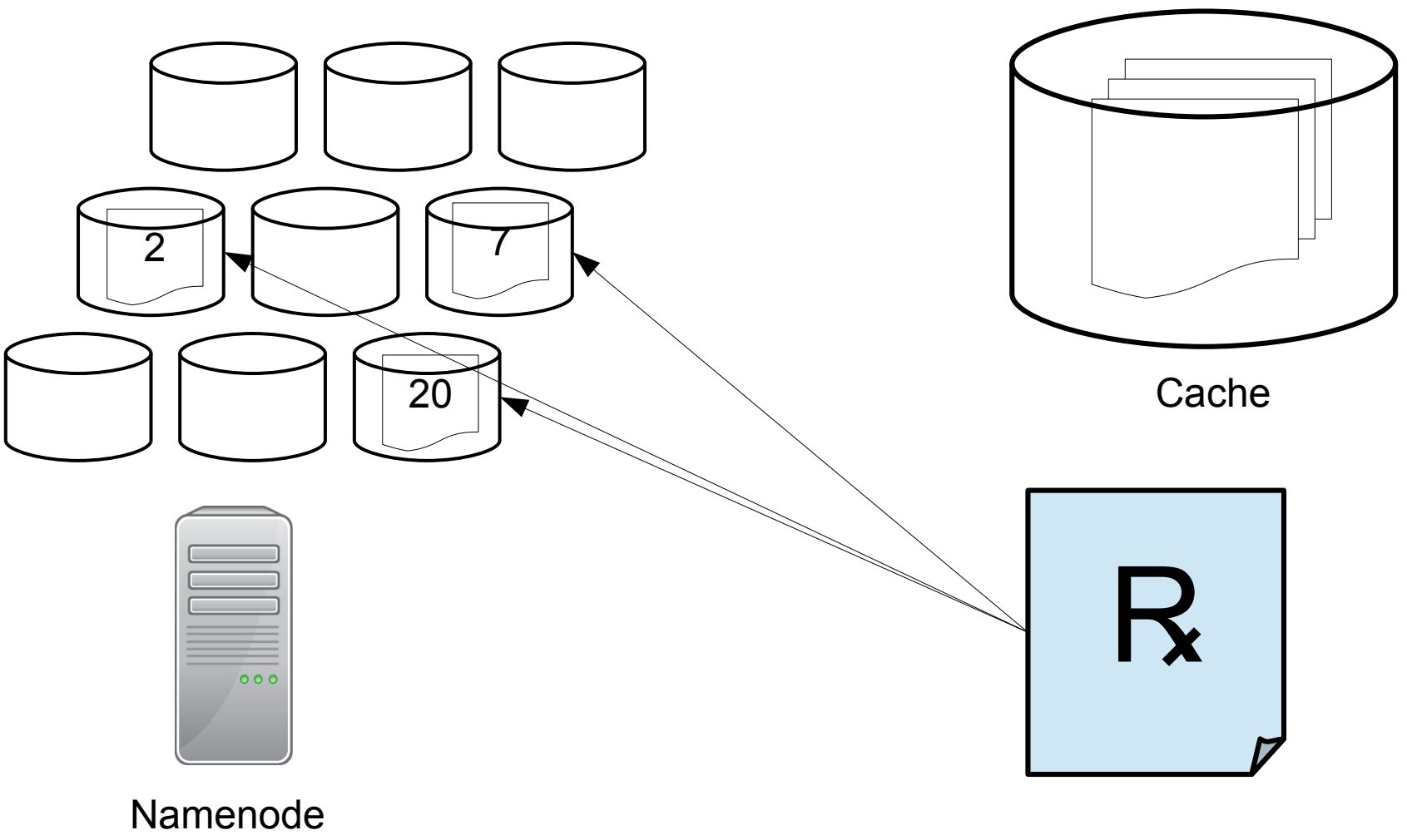




??!!



Healer Architecture



Running Fallback at UCSD

- In April 2014 we turned on hdfs-xrootd-fallback
- Since then we've reduced replication factor to 1 on a set of directories:

```
/cms/store/data/Run2012A
/cms/store/data/Run2012B
/cms/store/data/Run2012C
/cms/store/data/Run2012D
/cms/store/data/Fall13
/cms/store/data/Summer13
/cms/store/mc/Summer12_DR53X
/cms/store/himc
/cms/store/relval
```

- This has effectively freed 236 TB (11%) of physical storage space in our cluster

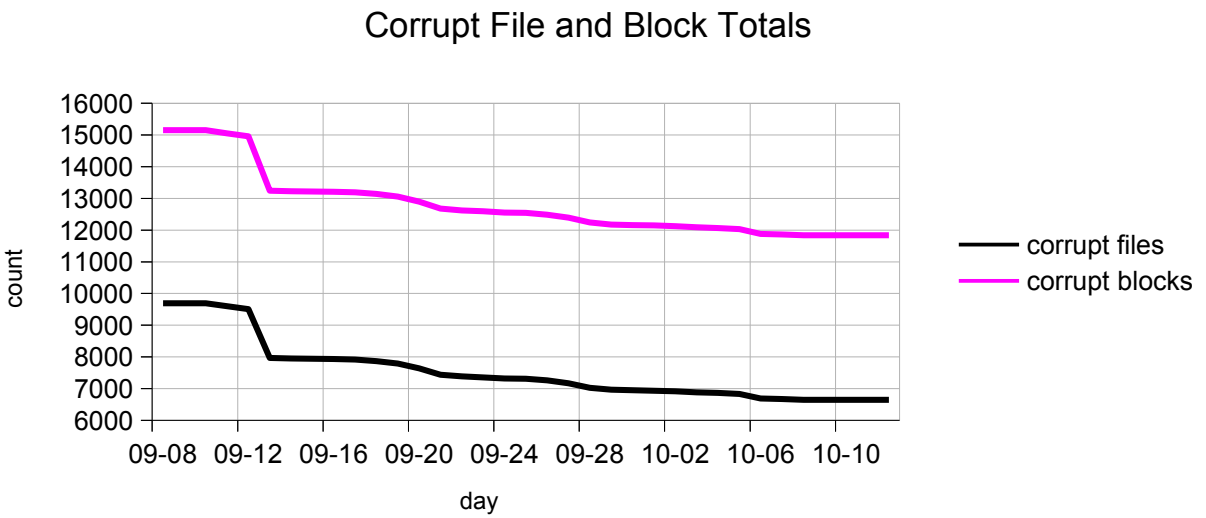
Running Fallback at UCSD

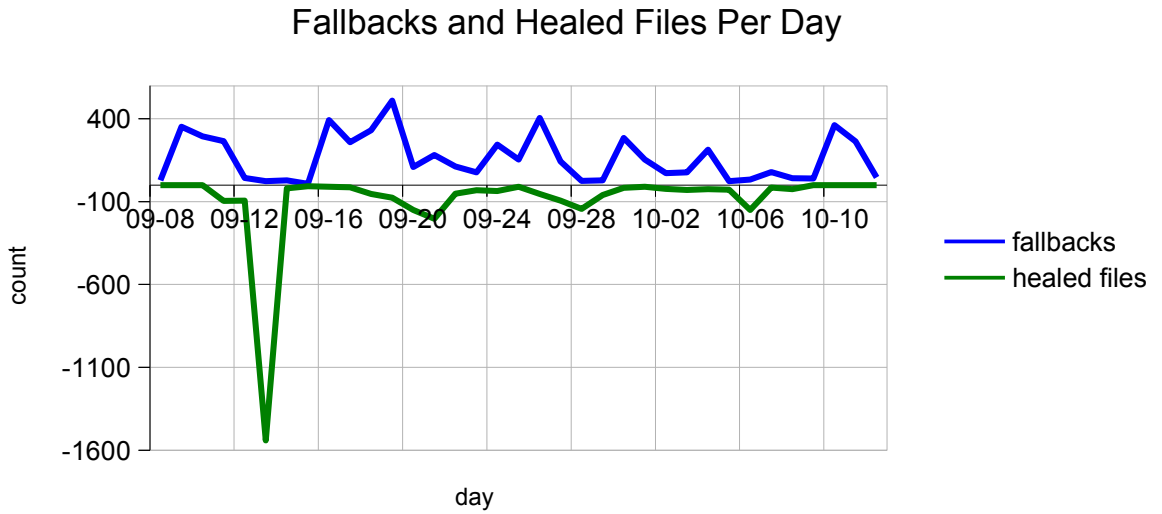
- Current storage summary:

Live Nodes	101
Configured Capacity	2.55 PB
DFS Used	1.96 PB
DFS Used%	76.82 %
Total Files	18173219
Total Blocks	43245486
Missing Blocks	15420

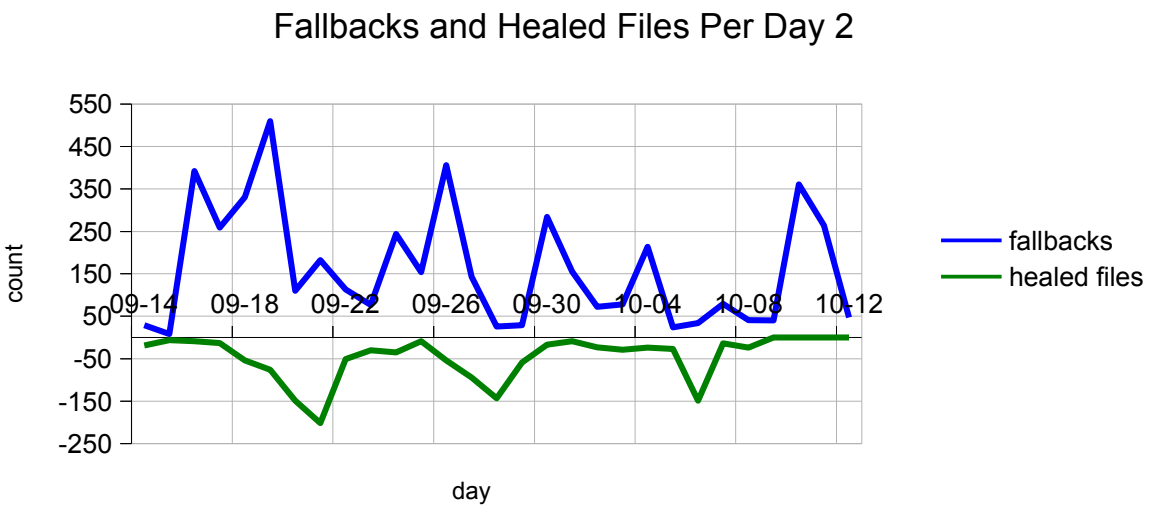
- On September 10, 2014, we finished implementing and turned on the hdfs-xrootd-healer

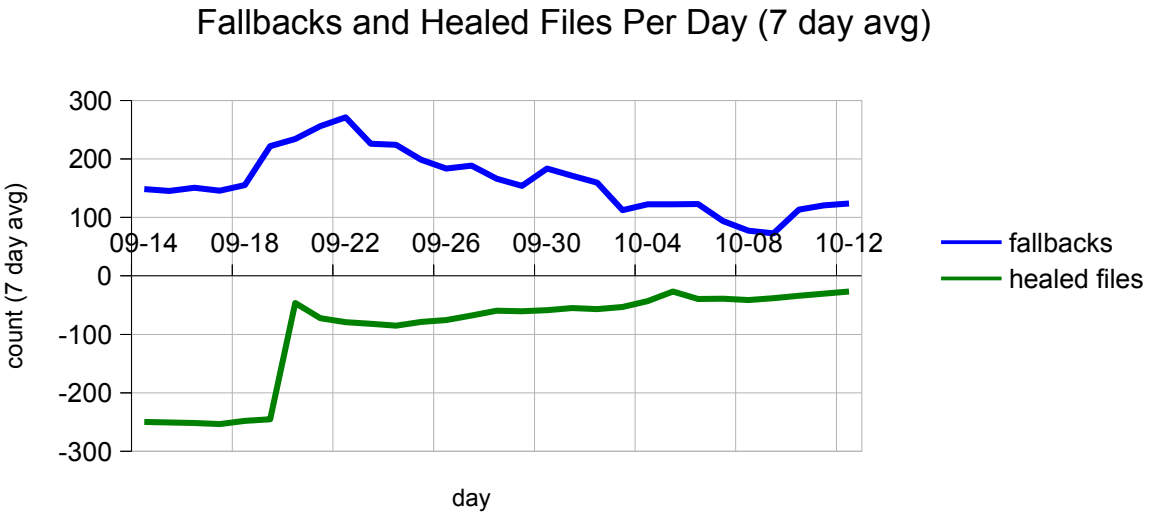
Fallback Statistics





Fallback Statistics





Data Corruption

- gridftp-hdfs plugin stores checksums in /cksum dir in hadoop (MD5,Adler32,CRC32,cksum)
- Performed bulk MD5 checksum verification over all healed files
- Of the 3048 files we healed, 41 were corrupt (1%), due to corrupt blocks received from XRootD
- We had to manually re-corrupt these blocks to ensure we don't serve corrupt data to users
- Turned off healing until we implement corruption protection
- Plan to contact sites that served corrupt blocks and work with admins to understand the source of corruption

Data Corruption

- Important finding – A subset of corrupt files went undetected when comparing Adler32 checksums!
- Errors were only detected using MD5
- These corrupt blocks have a strange but consistent pattern:

```
        offset          bits
orig 16974170 ...01100011
new  16974170 ...01100111
-------------------------
orig 16974186 ...01110101
new  16974186 ...01110001
```

- A single bit is flipped on, 3 positions from the right
- And then another is flipped off in the same position, 16 bytes later

Data Corruption

- Our best guess is this is due to some hardware malfunction, e.g. a bad network switch
- We have not yet determined if it is local (UCSD) or from a remote site serving us through XRootD

Summary of Fallback

	Count	Percent
Total Files	66410	100%
Good Files	51483	77%
Corrupt Files	11838	18%
Healed Files	3048	5%
Bad Cksum	41	0.06%