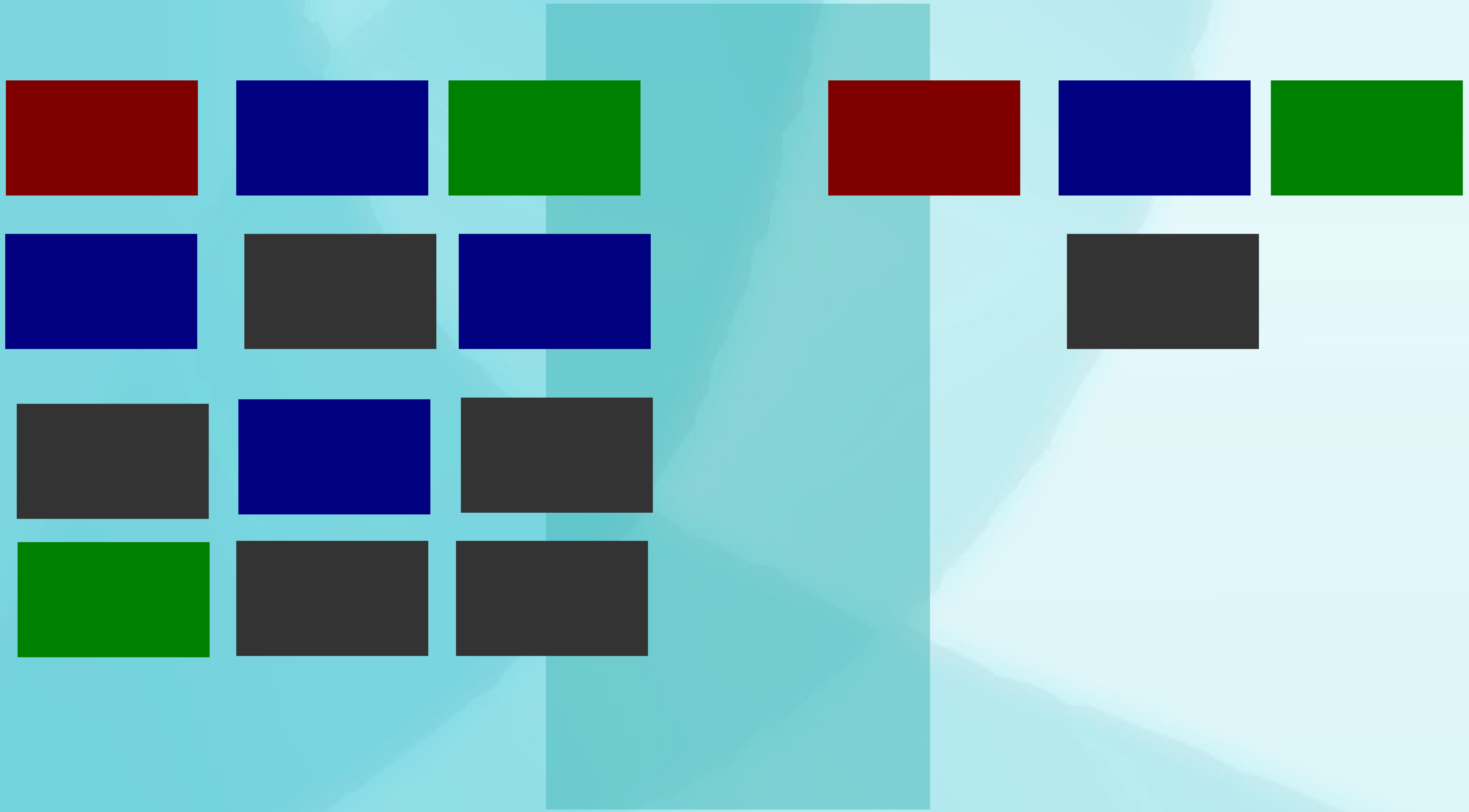
# OpenDedup

Open Source Block-Level Deduplication



## Copyright

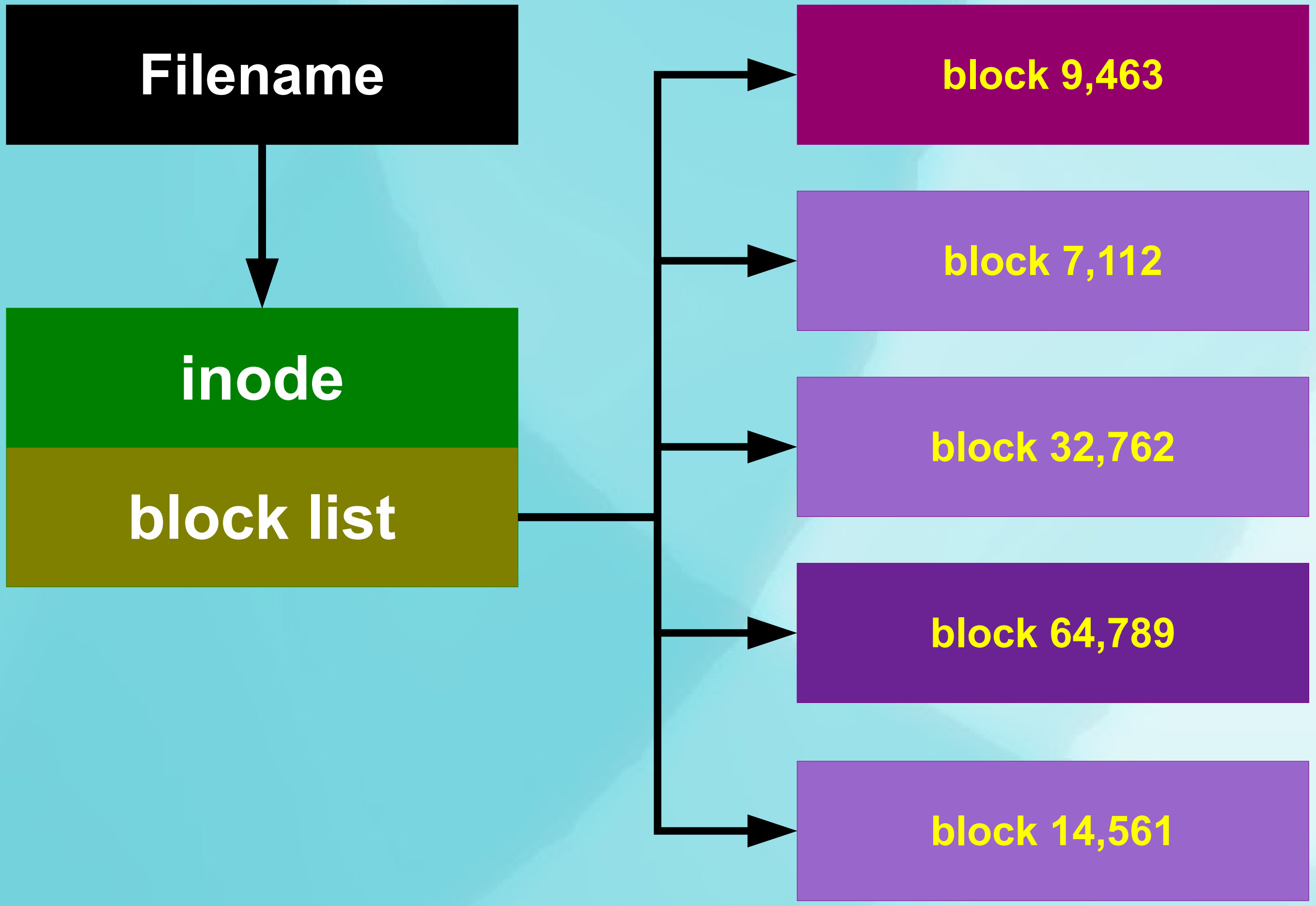
## The Proverb

“As storage becomes less and less expensive storage becomes more and more expensive.” --unknown

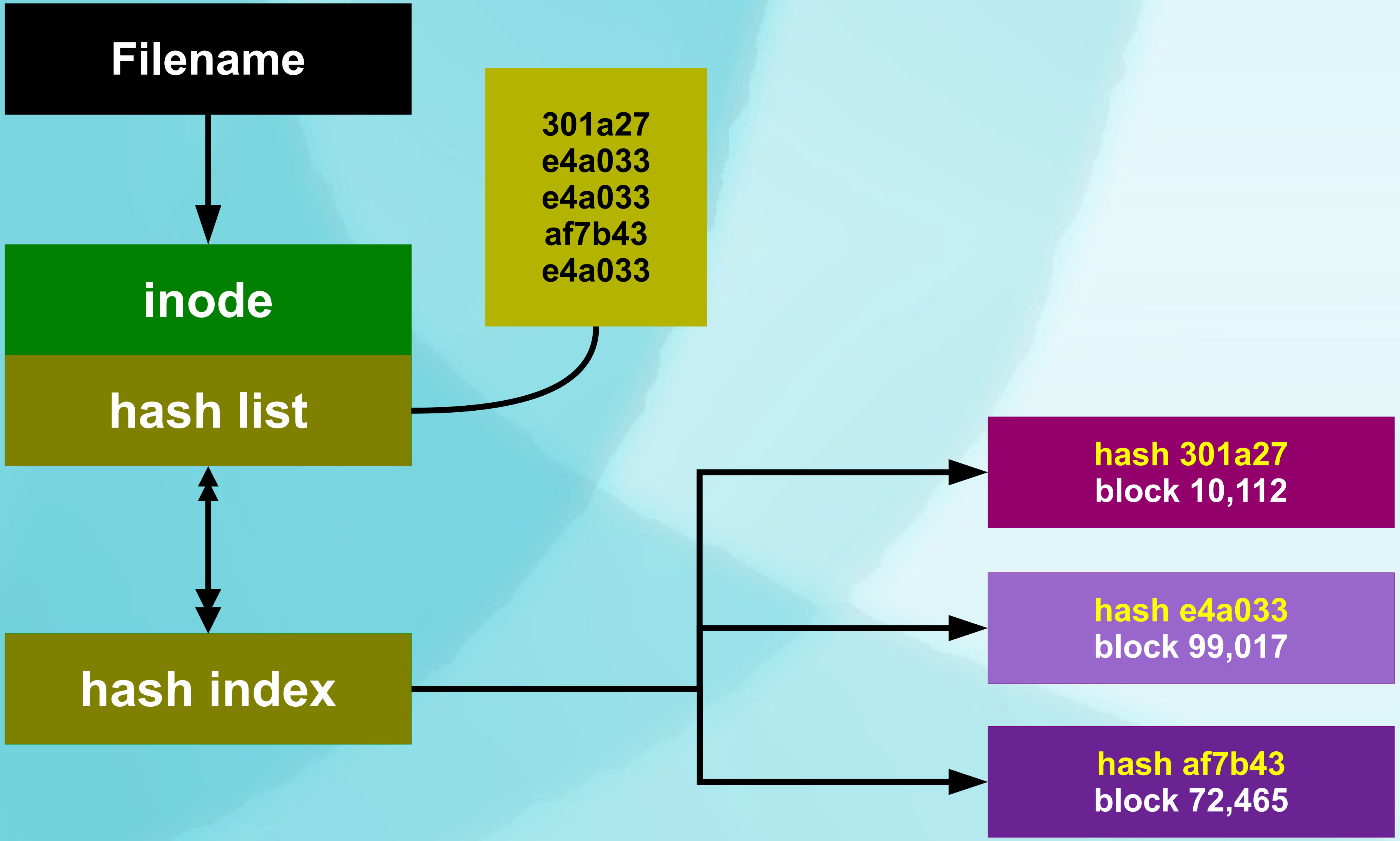
## Ways To Shrink Data

- Single-Instance Store
  - rsync with –link-dest
  - Cyrus IMAP singleinstancestore
- Instance Compression
  - zip
  - gzip
  - bzip
  - e[2/3]compr

# Traditional File-System



# Block Level Deduplication



## Block-Level Deduplication

- Reduced Storage Requirements
- Little impact on read performance
  - May actually be faster
- Transparent write operations
  - Cost can be off-loaded from the host
- No file-system meta-data bloat
  - A problem with rsync –link-dest
- Efficient Replication

# Block-Level Deduplication

- Everything hangs on the index
  - Corrupt the index, loose everything.
- Recovery next to impossible
  - Disk images are even less coherent.
- Memory Intensive
  - Requires very large hash-tables.
- Some CPU cost

## Types Of Data

- Things that de-duplicate well:
  - Backups
  - Virtual Disk Images (VMDKs)
  - Documents [doc/docx/odt/xls/ods/ppt/odp]
  - E-Mail
- Things that do not de-duplicate well:
  - Multimedia [music/video]
  - Compressed Images [JPEG/PNG/GIF]
  - Encrypted Data

# OPENDEDUP

- Also known as “SDFS”
- Supports volume replication
  - Distributed storage
  - RAIN
- Scalable
  - In excess of a petabyte of data
- File cloning
- Block sizes as small as 4K
- User space

## Requirements

- fuse 2.8 or greater
- Java 1.7
- attr

## Install

```
tar -C /opt -xzvf /tmp/sdfs-1.0.7.tar.gz
mv /opt/sdfs-bin /opt/sdfs
cd /opt/sdfs
tar xzvf /tmp/jre-7-fcs-bin-b147-linux-x64-27_jun_2011.tar.gz
ln -s jre1.7.0/ jre
export JAVA_HOME=/opt/sdfs/jre
export PATH=/opt/sdfs/jre/bin:/opt/sdfs:$PATH
export CLASSPATH=/opt/sdfs/lib
```

# Creating a local volume

```
mkfs.sdfs --volume-name=volume1 \
    --volume-capacity=100GB \
    --base-path=/srv/sdfs/volume1 \
    --io-chunk-size 64
```

```
--base-path <PATH>
--chunk-store-data-location <base-path/chunkstore/chunks>
--dedup-db-store <base-path/ddb>
--chunk-store-encrypt <true|false>
--chunk-store-local <true|false>
--io-chunk-size <SIZE in kB; use 4 for VMDKs, defaults to 128>
--volume-capacity <SIZE [MB|GB|TB]>
--volume-name <STRING>
```

# Mounting a volume

```
$ mount.sdfs -v volume1 -m /mnt1
Running SDFS Version 1.0.7
reading config file = /etc/sdfs/volume1-volume-cfg.xml
...
```

```
$ df -k
...
volume1-volume-cfg.xml
                    104857600         0 104857600   0% /mnt1
$ time cp -pvR /vms/Vista-100 .
...
`/vms/Vista-100/Primary Disk 001-000004-s029.vmdk' -> \
  `./Vista-100/Primary Disk 001-000004-s029.vmdk'
```

## Seeing the metadata

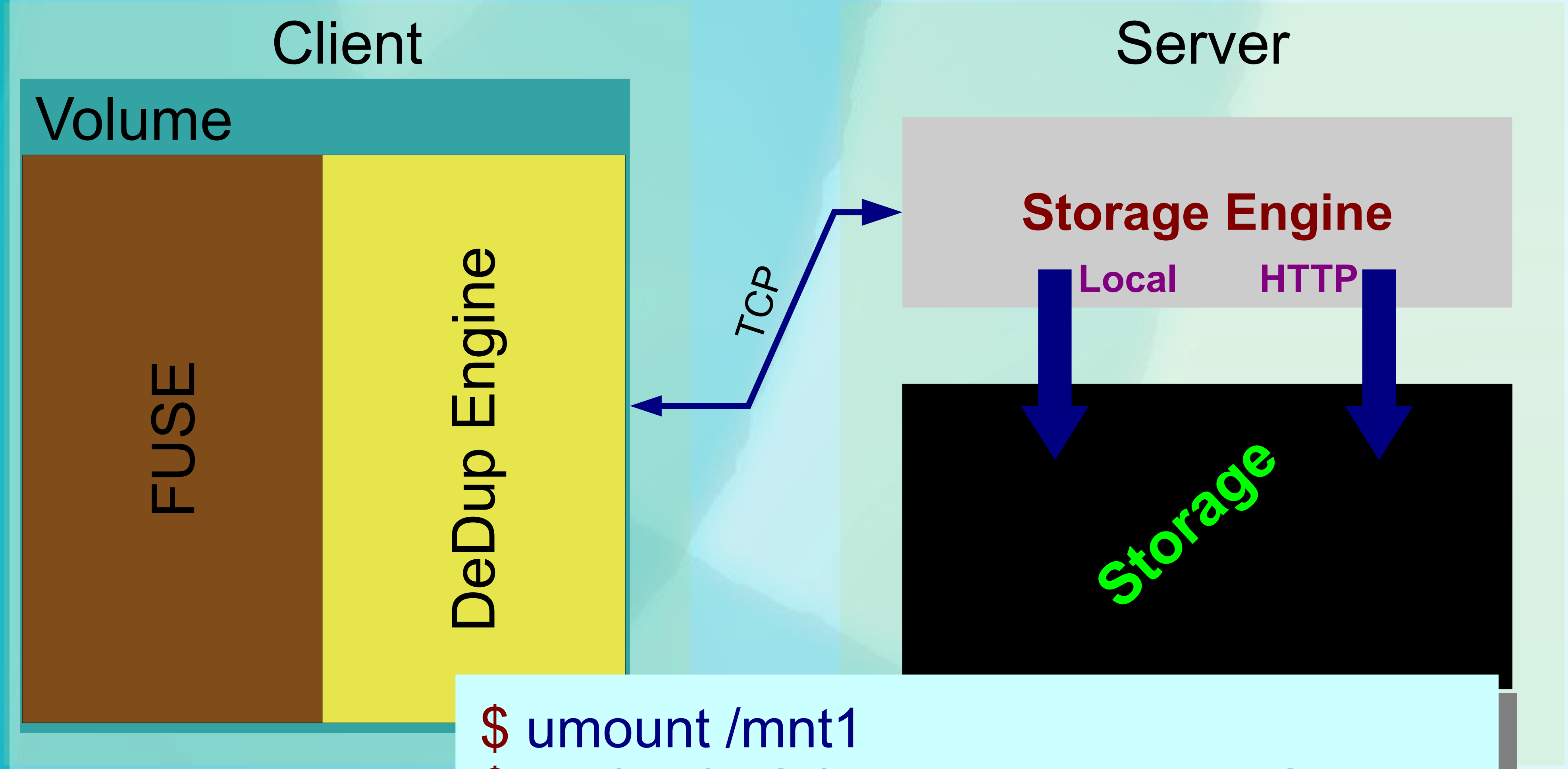
```
$ getfattr -d "Primary Disk 001-s001.vmdk"
# file: Primary Disk 001-s001.vmdk
user.dse.maxsize="107898470400"
user.dse.size="3519938560"
user.sdfs.ActualBytesWritten="1,523,056,640"
user.sdfs.BytesRead="0"
user.sdfs.DuplicateData="15,925,248"
user.sdfs.VMDK="false"
user.sdfs.VirtualBytesWritten="1,538,981,888"
user.sdfs.dedupAll="true"
user.sdfs.dfGUID="c39f5c25-328a-456b-a329-77f...
user.sdfs.file.isopen="true"
user.sdfs.fileGUID="75f9e4f4-56a4-48b0-a3a4-a2...
```

## Can we really use the metadata?

```
#!/usr/bin/env python
import xattr
f = open('/mnt1/Vista-100/Primary Disk 001-000003-s045.vmdk')
z = xattr.xattr(f)
for x, y in z.items():
    print x, y
```

```
user.sdfs.file.isopen false
user.sdfs.ActualBytesWritten 0
user.sdfs.VirtualBytesWritten 131072
user.sdfs.BytesRead 0
user.sdfs.DuplicateData 131072
user.dse.size 11560353792
user.dse.maxsize 107898470400
….
```

# Going network!



```
$ umount /mnt1
$ rm /etc/sdfs/volume1-volume-cfg.xml
$ rm -fR /srv/sdfs/*
```

# /etc/sdfs/hashserver-config.xml

```
<chunk-server>
  <network port="2222"
     hostname="0.0.0.0"
     use-udp="false"/>
  <locations
     chunk-store="/srv/sdfs/dchunks/chunkstore/chunks"
     hash-db-store="/srv/sdfs/ddb/hdb"/>
  <chunk-store pre-allocate="false"
     chunk-gc-schedule="0 0 0/2 * * ?"
     eviction-age="4"
     allocation-size="161061273600"
     page-size="4096"
     read-ahead-pages="8"/>
</chunk-server>
```

## Start Our Own Hash Server

```
$ export JAVA_HOME=/opt/sdfs/jre
$ export PATH=/opt/sdfs/jre/bin:/opt/sdfs:$PATH
$ export CLASSPATH=/opt/sdfs/lib
$ startDSEService.sh /etc/sdfs/hashserver-config.xml
```

```
$ netstat --listen --tcp --numeric –program
...
tcp        0      0 :::2222    :::*     LISTEN     13414/java
...
```

# Create a “remote” volume

```
$ mkfs.sdfs --volume-name=volume1 \
  --volume-capacity=100GB  \
--io-chunk-size 4
```

- Edit /etc/sdfs/volume1-volume-cfg.xml
  - Change “enable” attribute of the “local-chunkstore” element to “false”

```
<local-chunkstore allocation-size="107374182400" chunk-gc-schedule="0 0 0/4 * * ?"
chunk-store="/opt/sdfs/volumes/volume1/chunkstore/chunks" chunk-store-dirty-
timeout="1000" chunk-store-read-cache="5" chunkstore-
class="org.opendedup.sdfs.filestore.FileChunkStore" enabled="false" encrypt="false"
encryption-key="q@98IYEN@mqb6jkj2pV9gZIzSv3@WsUHh4J" eviction-age="6" gc-
class="org.opendedup.sdfs.filestore.gc.PFullGC" hash-db-
store="/opt/sdfs/volumes/volume1/chunkstore/hdb" pre-allocate="false" read-ahead-
pages="8"/>
```

# /etc/sdfs/routing-config.xml

```
<routing-config><servers>
  <server name="server1" host="127.0.0.1" port="2222"
      enable-udp="false" compress="false"
      network-threads="8"/>
  <server name="server2" host="127.0.0.1" port="2222"
    enable-udp="false" compress="false"
    network-threads="8"/>
</servers><chunks>
<chunk name="00" server="server1"/>
<chunk name="01" server="server1"/>
…
<chunk name="fe" server="server2"/>
<chunk name="ff" server="server2"/>
</chunks></routing-config>
```

## Mount our remote store

```
$ mount.sdfs -r /etc/sdfs/routing-config.xml \
  -v volume1 -m /mnt1
```

```
$ df -k
...
volume1-volume-cfg.xml
                  104857600         0 104857600   0% /mnt1
$ cp -pvR /iso/Heretic2.iso /mnt1/Heretic2-1.iso
$ cp -pvR /iso/Heretic2.iso /mnt1/Heretic2-1.iso
```

## Do we have two copies?

```
$ getfattr -d /mnt1/Heretic2-1.iso
…
user.sdfs.ActualBytesWritten="242,933,760"
user.sdfs.DuplicateData="335,872"
user.sdfs.VirtualBytesWritten="243,269,632"
...
```

```
$ getfattr -d /mnt1/Heretic2-1.iso
…
user.sdfs.ActualBytesWritten="0"
user.sdfs.DuplicateData="243,269,632"
user.sdfs.VirtualBytesWritten="243,269,632"
...
```

# The server's chunkstore

```
$ cd /srv/sdfs
$ du -ks *
10,670,192  dchunks
184,972  ddb
$ ls -l dchunks/chunkstore/chunks/
trw-r--r-- 10,915,586,048 Jul 27 17:11 chunks.chk
$ ls -l ddb/hdb/
-rw-r--r--    189,210,598 Jul 27 17:11 hashstore-sdfs
```

# Recommendations

- Memory
  - 2GB allocation OK for:
    - 200GB@4KB chunks
    - 6TB@128KB chunks
  - Edit mount.sdfs/startDSE.sh to increase
    - Change this: "*-Xmx2g*"
    - Each chunk requires 25bytes
    - *footprint* = (*volume* / *chunk-size*) * 25

32TB of data at 128KB requires 8GB of RAM. 1TB @ 4KB equals the same 8GB.

## Janitorial Jobs

- How do chunks gets eliminated?
  - FUSE tells the DSE what blocks are in use.
  - DSE checks for unclaimed blocks.
    - Every four hours.
    - For 8 hours upon mount.
  - Blocks unclaimed for 10 hours released.
  - Configuration options:
    - FUSE: claim-hash-schedule
    - DSE: chunk-gc-schedule
    - Both must be more frequent than eviction-age.

## Calling the janitor

- A chunk-store cleaning can be manually requested.

```
$ setfattr -n user.cmd.cleanstore \
    -v 5555:15
    /var/lib/pgsql
```

- Many parameters can be tweaked via setfattr.
  - Deduplication can be disabled on specific files.

```
$ setfattr -n user.cmd.dedupAll \
    -v 556:false <path to file>
```

## Making a cloud drive

On Amazon S3

```
$ mkfs.sdfs  --volume-name=<volume name> \
    --volume-capacity=<volume capacity> \
    --aws-enabled=true \
    --aws-access-key=<the aws assigned access key> \
    --aws-bucket-name=<bucket name> \
    --aws-secret-key=<assigned aws secret key> \
    --chunk-store-encrypt=true
$ mount.sdfs <volume name> <mount point>
```