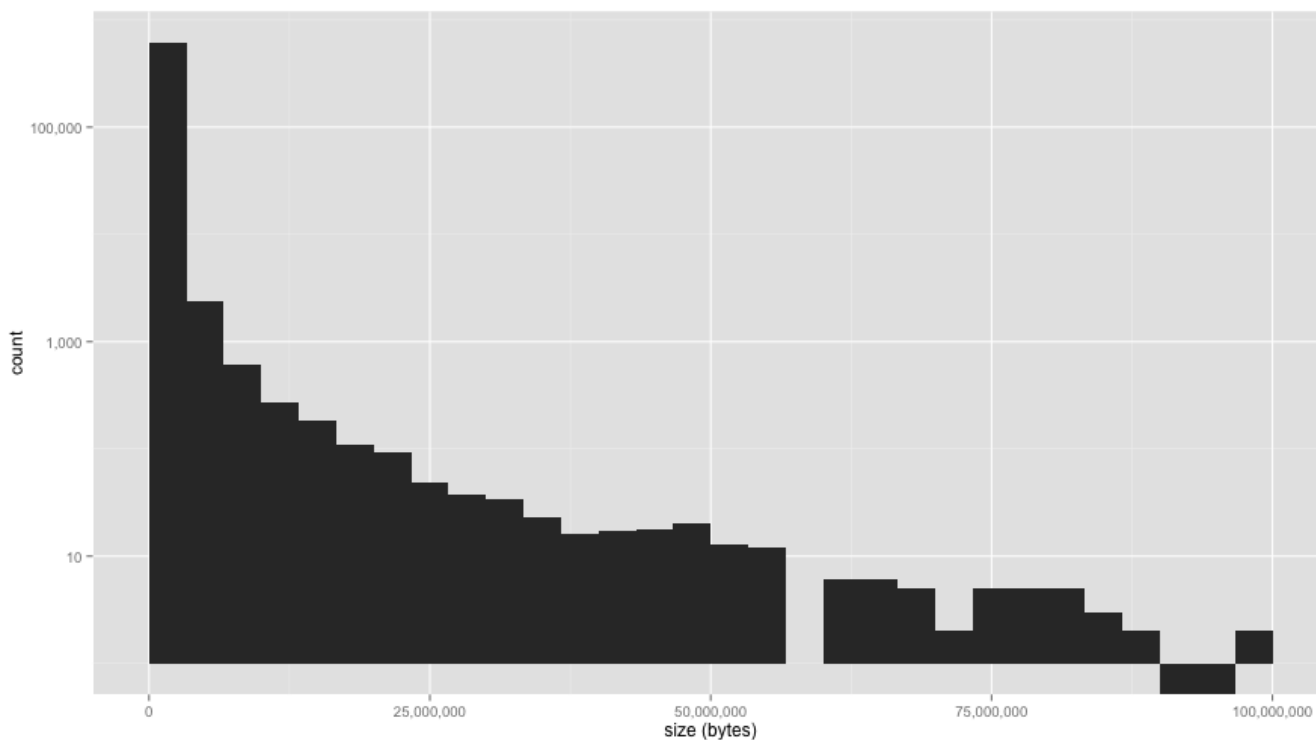


Stanford SALT Collection

- [Data Overview](#)
 - [Test 1: Simple Ingest into Fedora 3](#)
 - [Test 1a: Single-threaded ingest](#)
 - [Test 1b: Single-threaded iteration](#)
 - [Test 1c: 8-thread ingest test](#)
 - [Test 1d: Multi-threaded iteration test](#)
 - [Test 2: Simple Ingest into Fedora 4](#)
 - [Test 2a: Ingest all the data as containers and binaries, one at a time](#)
 - [Test 2b: Ingest all the data as containers arranged in a druid tree](#)
 - [Test 2c: Ingest all the data as containers in a druid tree AND use for:batch](#)
 - [Test 2d: Use a 4-node cluster to do a druid-tree ingest](#)
 - [Test 3: Realistic Ingest into Fedora 3](#)
 - [Test 4: Realistic Ingest into Fedora 4](#)

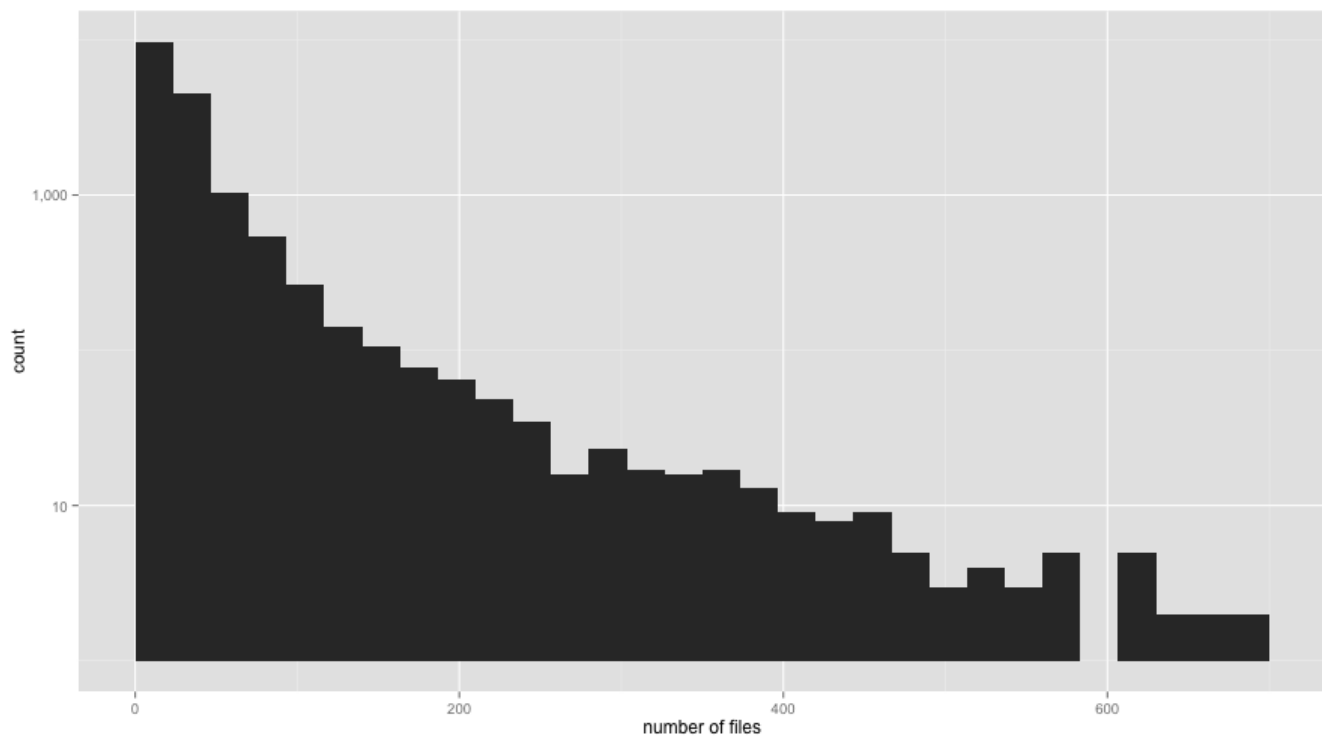
Data Overview

Stanford has a collection of publications consisting of page images, metadata, and arrangement ([Saltworks](#)), containing 16712 objects/655237 items /273GB of data with the following distribution:



```
> quantile(file_sizes$size, c(0, .5, .7, .9, 1))
0% 50% 70% 90% 100%
0 43447 195719 1835010 288032768

> quantile(object_size$V3, c(0, .5, .7, .9, .95, .99, 1))
0% 50% 70% 90% 95% 99% 100%
51302 4680240 10743517 39806094 72375144 221829327 1230705287
```



```
> quantile(file_counts$X1, c(0, .5, .7, .9, .95, .99, 1))
 0%    50%    70%    90%    95%    99%   100%
7.00  22.00  29.00  62.00  99.40 280.08 1478.00
```

In production, the object metadata is stored in Fedora, but the page images and other assets are stored on the file system and (somehow associated back to the object.. TBD).

Objects contain the following datastreams:

Datastream ID	MIME Type
DC	text/xml
RELS-EXT	text/xml
extracted_entities	application/xml
location	text/xml
zotero	application/rdf+xml

On the filesystem are a variety of files (including some duplicates of data in fedora?), e.g.:

- 4.0K DC
- 20K Feigenbaum_00013946-METS.xml
- 4.0K Feigenbaum_00013946-TEXT.xml
- 4.0K RELS-EXT
- 44K bd826tf2716.pdf
- 4.0K bd826tf2716.txt
- 72K bd826tf2716_00001.jp2

- 8.0K bd826ff2716_00001.xml
- 64K bd826ff2716_00002.jp2
- 8.0K bd826ff2716_00002.xml
- 68K bd826ff2716_000BW.jp2
- 4.0K checksum
- 4.0K descMetadata
- 4.0K extracted_entities.xml
- 4.0K flipbook.json
- 4.0K flipbook.old
- 0 location
- 0 properties
- 0 stories
- 4.0K thumb.jpg
- 4.0K zotero.xml

Test 1: Simple Ingest into Fedora 3

For a first test, we're going to ingest all the data from the filesystem into a clean fcrepo3 repository, using the filename as the datastream name.

Using Fedora 3.7.1, clean install, using these properties:

```
database=mysql
database.driver=included
database.jdbcDriverClass=com.mysql.jdbc.Driver
database.mysql.jdbcDriverClass=com.mysql.jdbc.Driver
database.mysql.driver=included
database.jdbcURL=jdbc:mysql://localhost/fedora?useUnicode=true
database.mysql.jdbcURL=jdbc:mysql://localhost/fedora?useUnicode=true
database.username=fedora
database.password=redacted
install.type=custom
deploy.local.services=false
install.tomcat=false
servlet.engine=existingTomcat
fedora.home=/home/lyberadmin/apps/fedora/home
fedora.serverHost=sul-fedora-dev-a.stanford.edu
fedora.serverContext=fedora
tomcat.http.port=8080
tomcat.shutdown.port=8005
ssl.available=true
tomcat.ssl.port=8443
tomcat.home=/usr/share/tomcat6
ri.enabled=true
messaging.enabled=false
messaging.uri=
apim.ssl.required=false
apia.ssl.required=false
apia.auth.required=false
fesl.authz.enabled=false
fesl.authn.enabled=true
xacml.enabled=false
keystore.file=included
```

Tomcat is proxied through an Apache HTTPD server.

Using bash:

```
#!/bin/bash
base_url="http://fedoraAdmin:fedoraAdmin@localhost/fedora"

RuntimePrint()
{
    duration=$(echo "scale=3; (${m2t}-${m1t})/(1*10^09)"|bc|sed 's/^\.0./')
    echo -e "${objectId} ${datastreams} ${size} ${duration}\tsec"
    echo -e "${objectId} ${datastreams} ${size} ${duration}" >> /data/fcrepo3-total-create-object-time
}

CreateObject() {
    pid="druid:$1"
    curl -X POST "$base_url/objects/$pid" &> /dev/null
    cd /data-ro/assets/$1

    for f in $( ls ); do
        datastreams=${datastreams+1}
        size=${size+`stat -c "%s" $f`}
        curl -X POST --data-binary @$f "$base_url/objects/$pid/datastreams/$f?controlGroup=M" &> /dev/null
    done
    cd /data
}

BenchmarkObject() {
    objectId=$1
    if [ -d /data-ro/assets/$objectId ]; then
        m1t=$(date +%s%N); m1l=$LINENO
        CreateObject $objectId
        m2t=$(date +%s%N); m2l=$LINENO; RuntimePrint
    fi
}

export -f BenchmarkObject
export -f CreateObject
export -f RuntimePrint
export base_url

cat - | parallel -P $THREADS --env _ BenchmarkObject
```

Test 1a: Single-threaded ingest

```
> quantile(create$V4, c(0, .5, .7, .9, .95, .99, 1))
 0%      50%      70%      90%      95%      99%     100%
0.39300  1.46300  2.31500  6.64700 11.65780 36.77232 353.48700
```

0.2597 objects/s (objects per second)

Test 1b: Single-threaded iteration

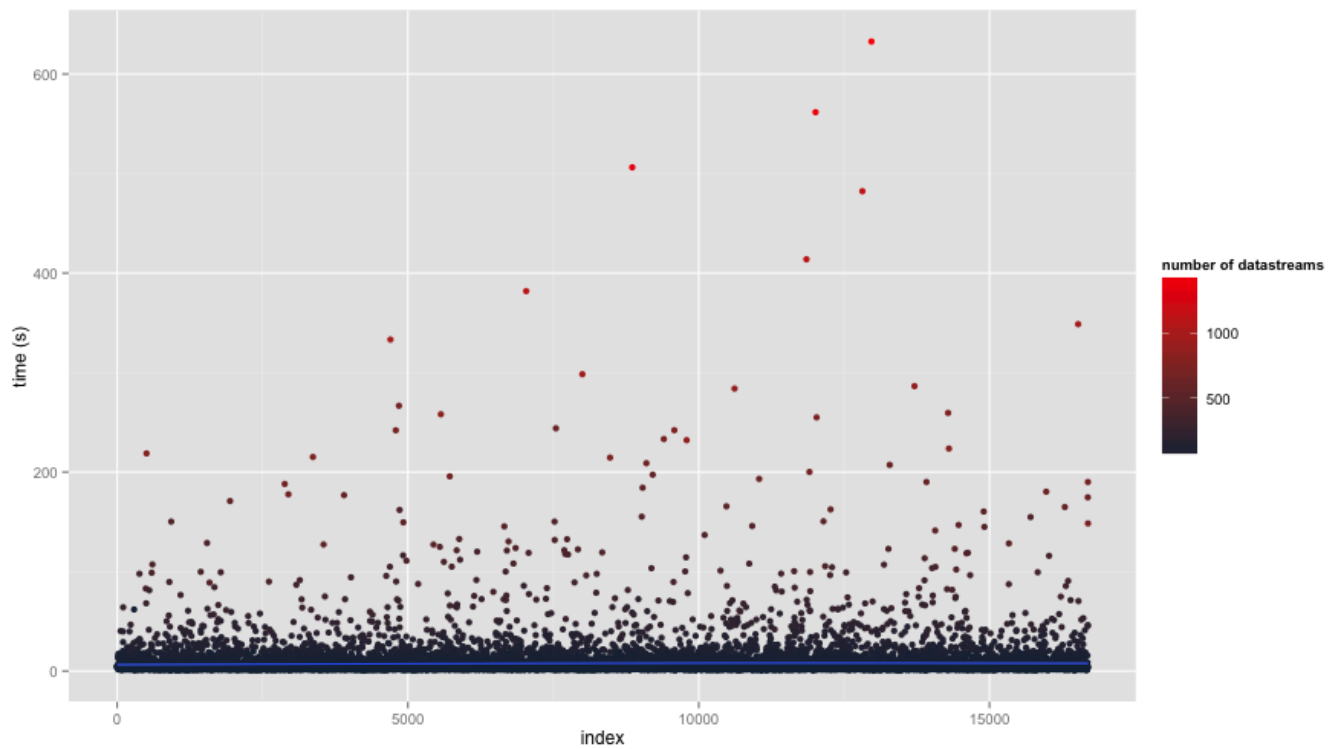
Retrieve object profile

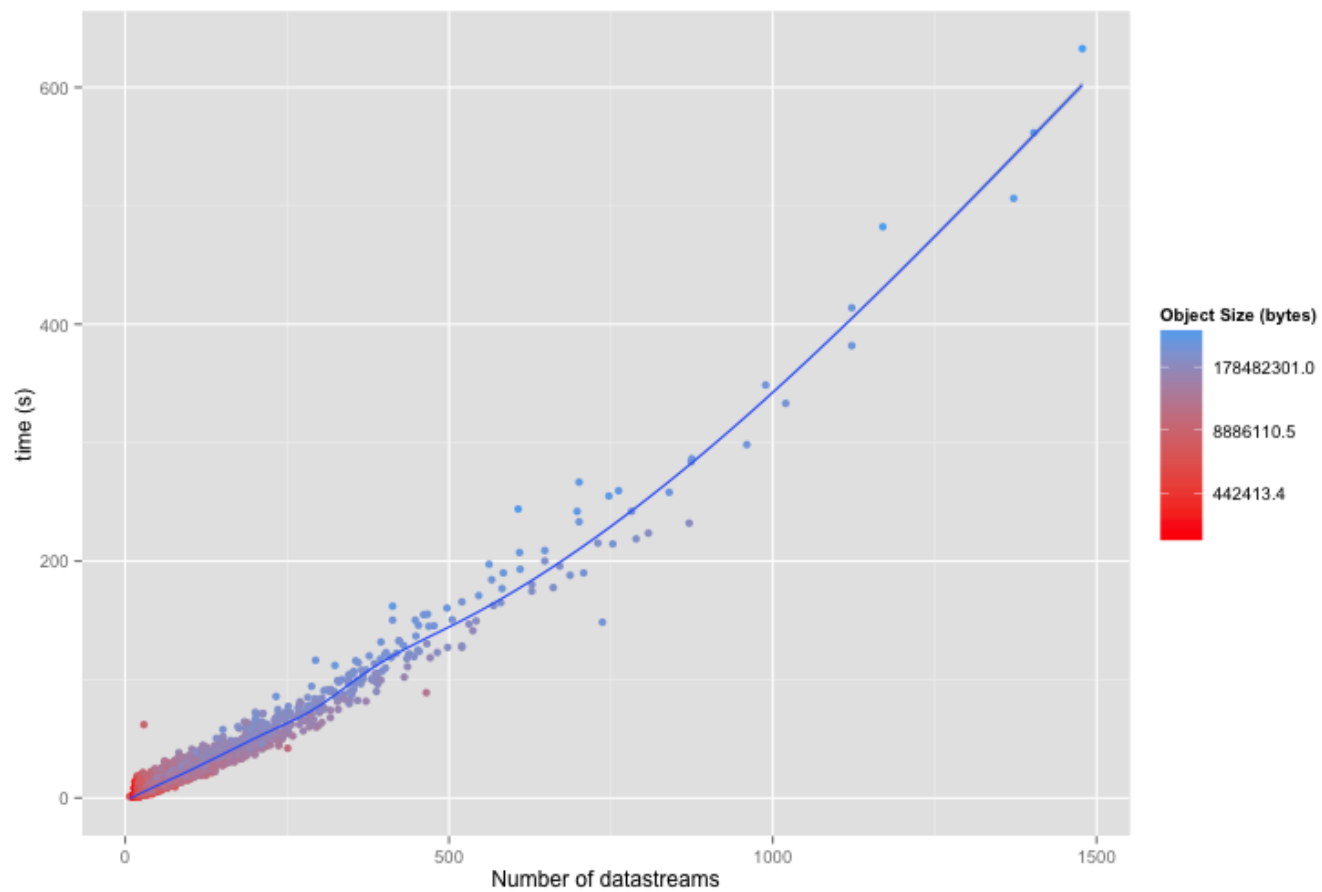
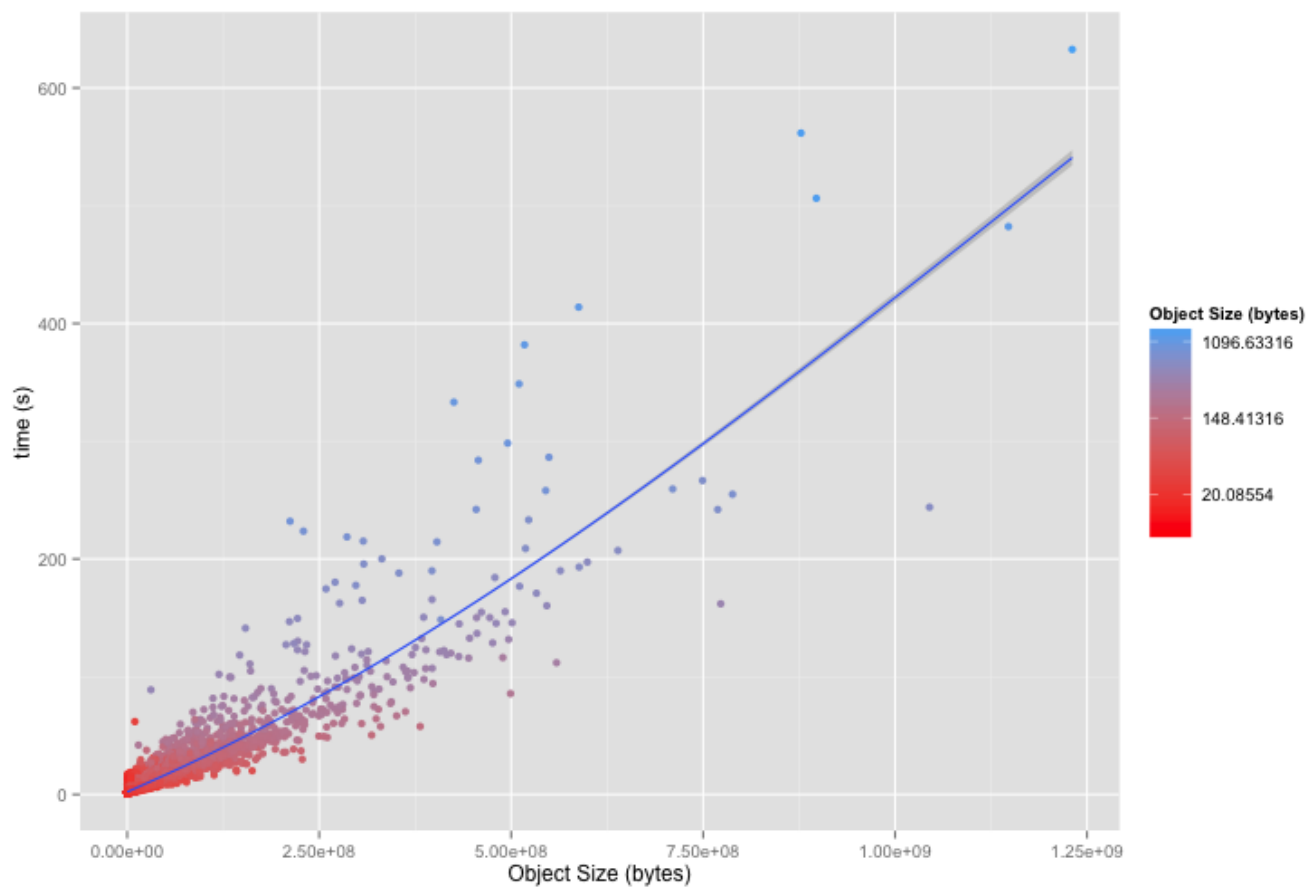
```
> quantile(data$V2, c(0, .5, .7, .9, .95, .99, 1))
 0%   50%   70%   90%   95%   99%  100%
0.002 0.054 0.062 0.077 0.089 0.123 3.580
```

Test 1c: 8-thread ingest test

```
> quantile(create$V4, c(0, .5, .7, .9, .95, .99, 1))
 0%      50%      70%      90%      95%      99%     100%
0.71400  3.46600  5.09440 14.32520 23.55560 72.11128 632.71200
2206.24user 5396.22system 4:29:58elapsed 46%CPU (0avgtext+0avgdata 1133728maxresident)k
584337920inputs+3216976outputs (67566major+823430581minor)pagefaults 0swaps
Tue Nov 19 19:09:08 PST 2013 : 16693 objects

1.031 objects/s (objects per second)
```





Test 1d: Multi-threaded iteration test

```
4 threads:
> quantile(read$V2, c(0, .5, .7, .9, .95, .99, 1))
 0%   50%   70%   90%   95%   99%  100%
0.011 0.013 0.014 0.017 0.020 0.031 0.073
Tue Nov 19 14:08:44 PST 2013 : retrieving all objects
160.65user 247.90system 2:42.00elapsed 252%CPU (0avgtext+0avgdata 40736maxresident)k
0inputs+267608outputs (0major+63796227minor)pagefaults 0swaps
Tue Nov 19 14:11:26 PST 2013 : 16693 objects
Tue Nov 19 14:11:26 PST 2013 : done
103 objects/s (objects per second)

8 threads:
> quantile(read$V2, c(0, .5, .7, .9, .95, .99, 1))
 0%   50%   70%   90%   95%   99%  100%
0.011 0.022 0.025 0.031 0.034 0.045 0.093
Tue Nov 19 14:05:10 PST 2013 : retrieving all objects
159.54user 251.87system 2:28.86elapsed 276%CPU (0avgtext+0avgdata 40880maxresident)k
0inputs+267608outputs (0major+63891890minor)pagefaults 0swaps
Tue Nov 19 14:07:39 PST 2013 : 16693 objects
Tue Nov 19 14:07:39 PST 2013 : done
112.1 objects/s (objects per second)

16 threads:
> quantile(read$V2, c(0, .5, .7, .9, .95, .99, 1))
 0%   50%   70%   90%   95%   99%  100%
0.012 0.024 0.028 0.035 0.040 0.052 0.149
Tue Nov 19 14:11:50 PST 2013 : retrieving all objects
161.64user 264.68system 2:30.56elapsed 283%CPU (0avgtext+0avgdata 41104maxresident)k
0inputs+267608outputs (0major+64217330minor)pagefaults 0swaps
Tue Nov 19 14:14:20 PST 2013 : 16693 objects
Tue Nov 19 14:14:20 PST 2013 : done
110.9 objects/s (objects per second)
```

Test 2: Simple Ingest into Fedora 4

Ingest all the data into fcrepo4 as [binaries](#) on [containers](#).

Using jgroups configuration at <https://gist.github.com/cbeer/fd3997e40fe014eab071>

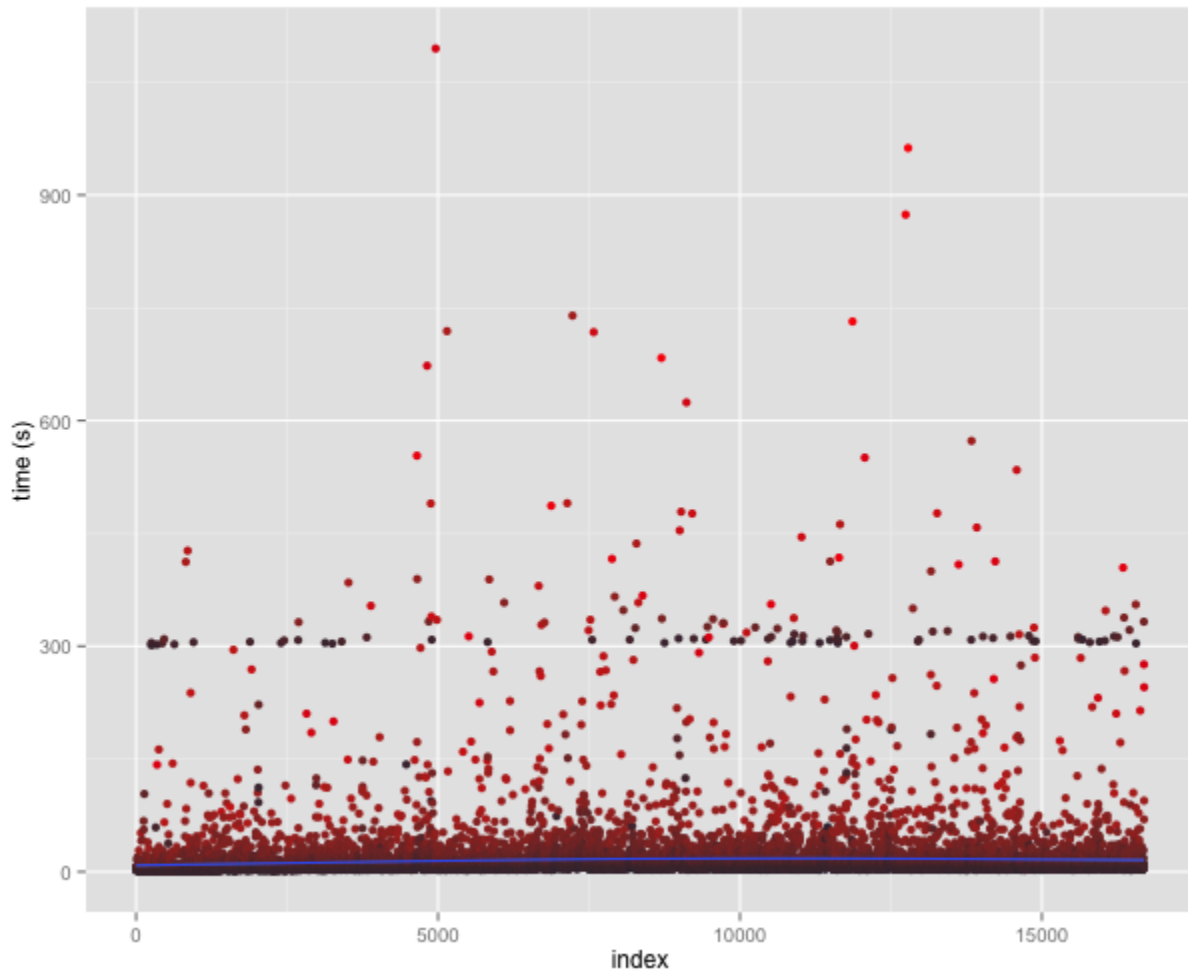
Using curl:

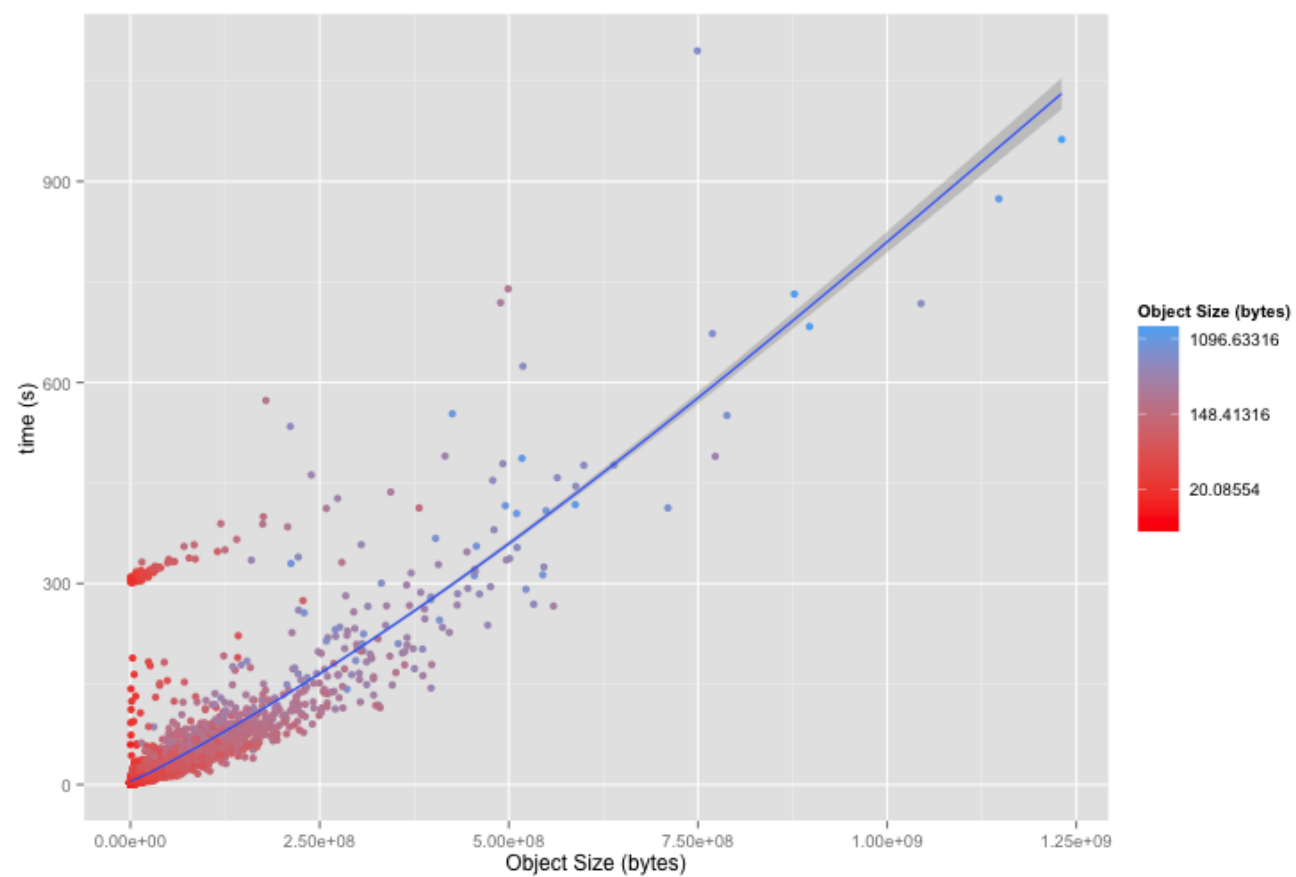
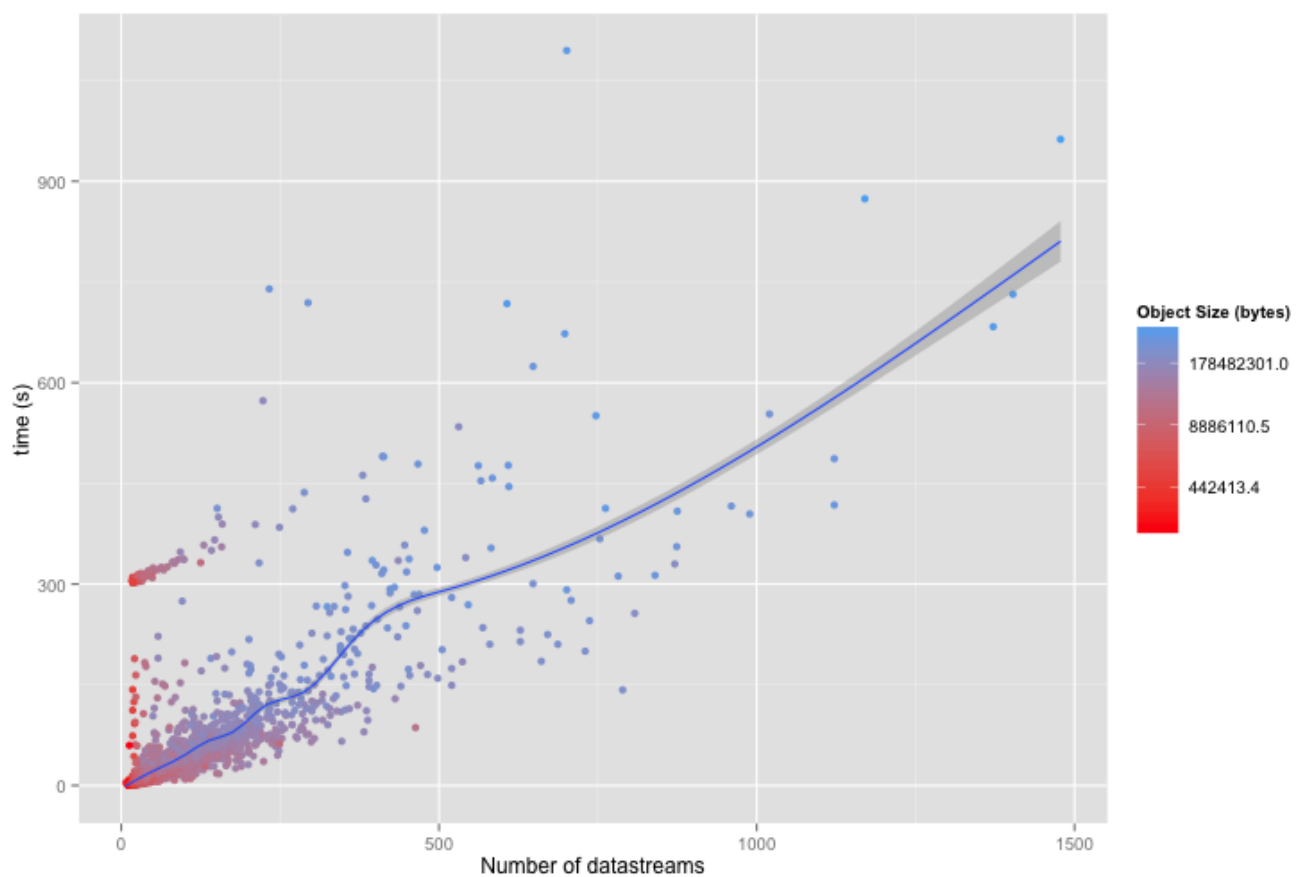
Test 2a: Ingest all the data as containers and binaries, one at a time

Test 2b: Ingest all the data as containers arranged in a druid tree

```
> quantile(create$V4, c(0, .5, .7, .9, .95, .99, 1))
 0%   50%   70%   90%   95%   99%  100%
0.6590   6.1800   9.1844  24.7972  44.7202 226.7644 1094.8120
```

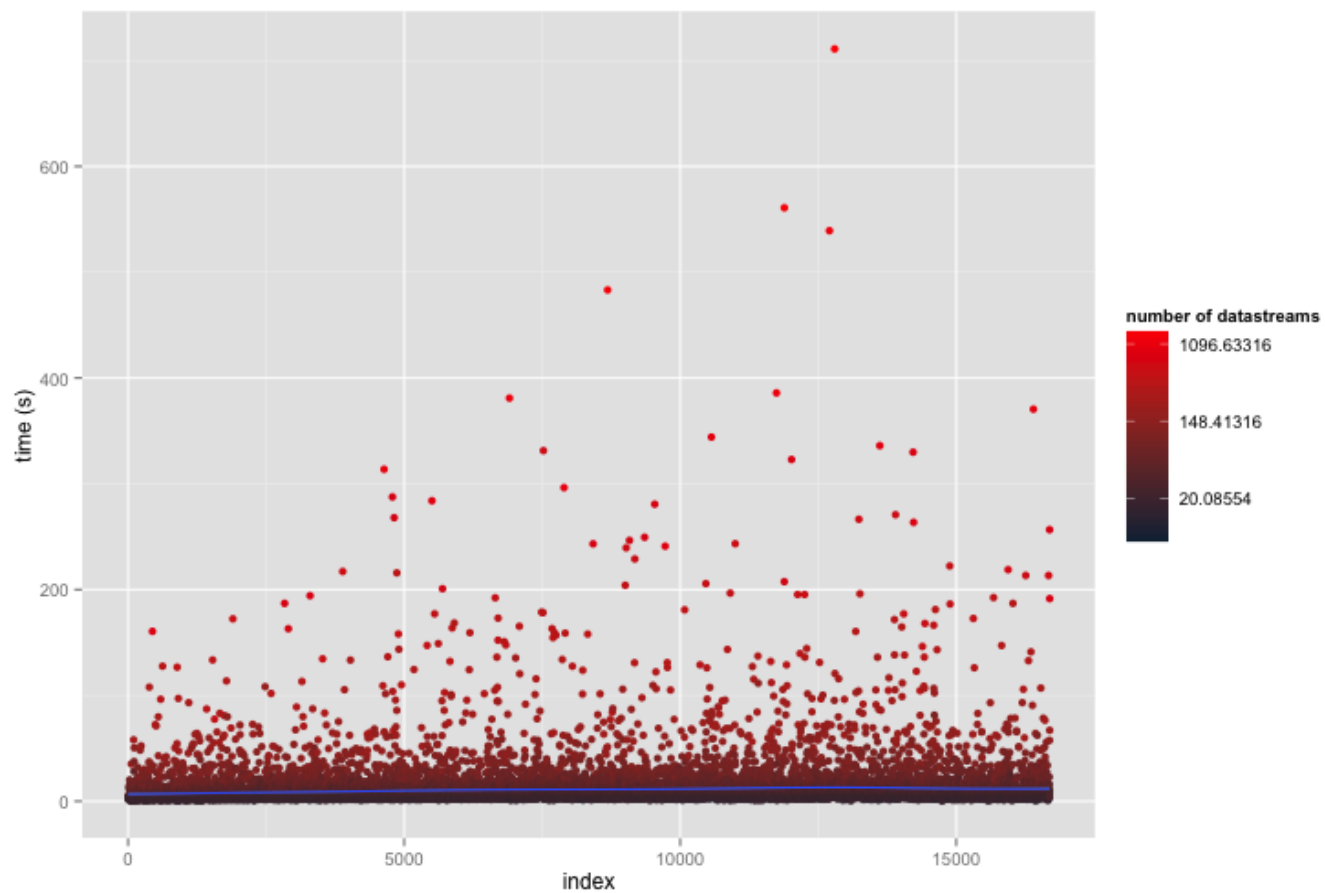
Ingest speed over time

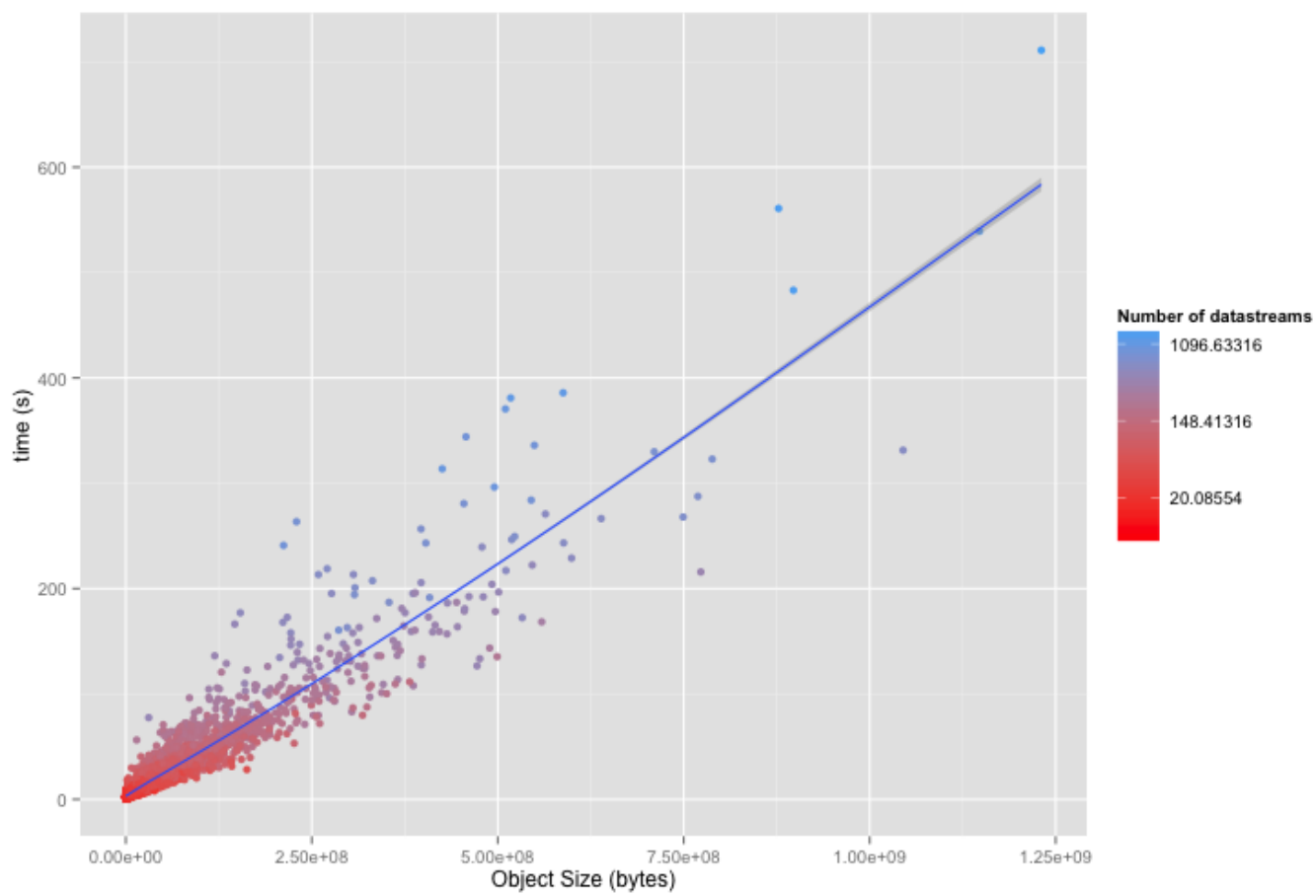
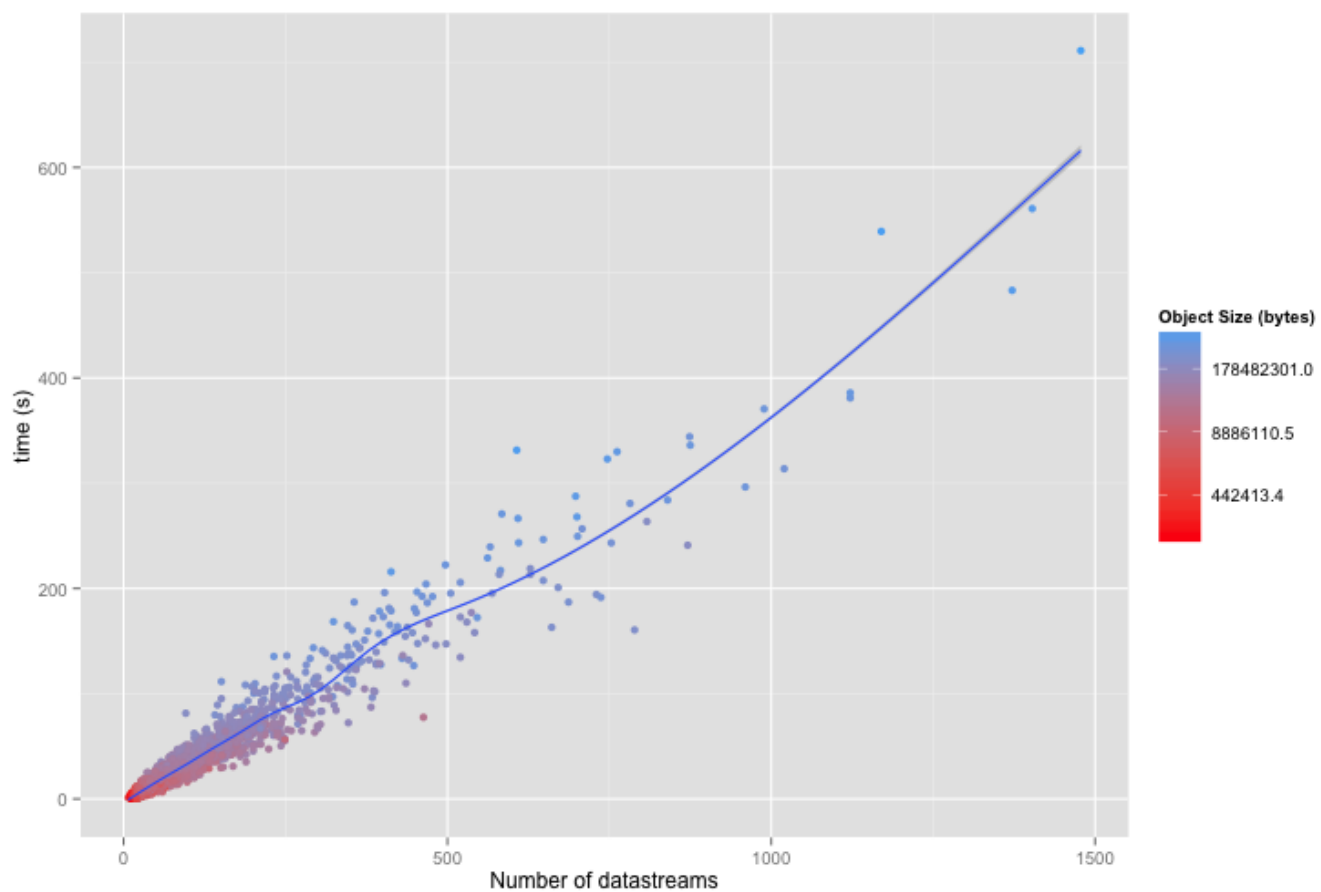




Test 2c: Ingest all the data as containers in a druid tree AND use fcr:batch

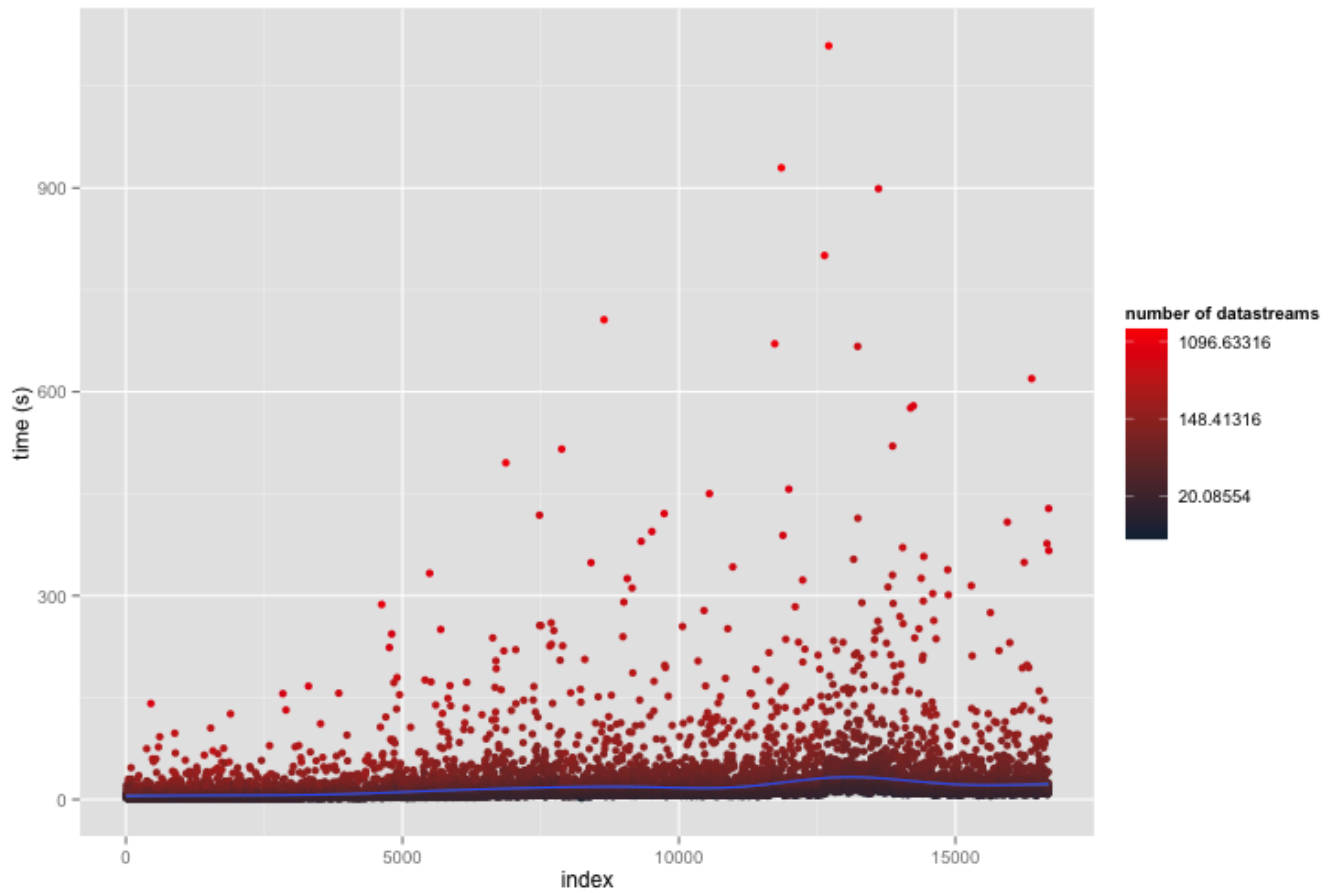
```
> quantile(create$V4, c(0, .5, .7, .9, .95, .99, 1))
 0%    50%    70%    90%    95%    99%   100%
0.4670  5.2440  7.8554 20.3558 33.8186 101.2618 711.0130
```

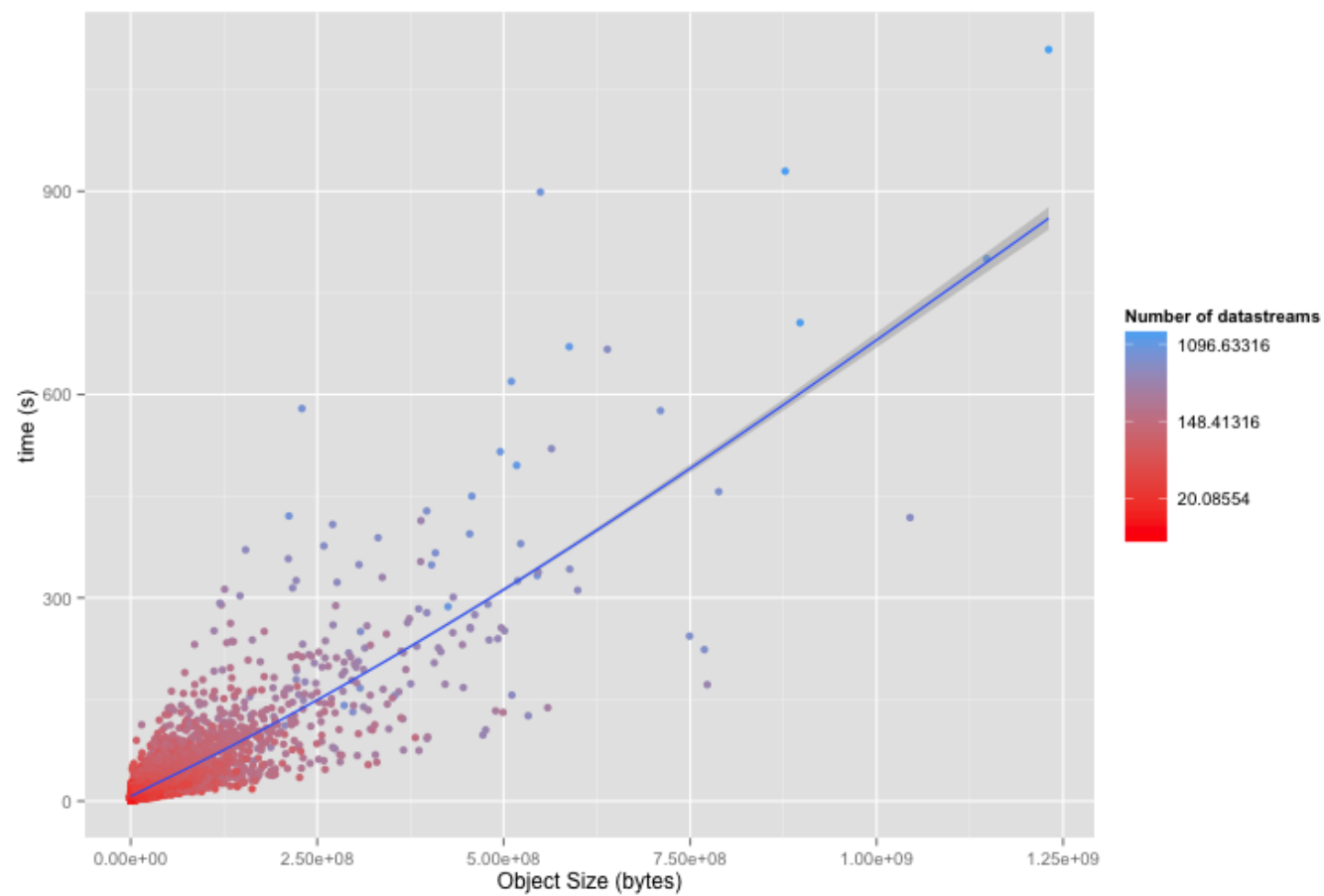
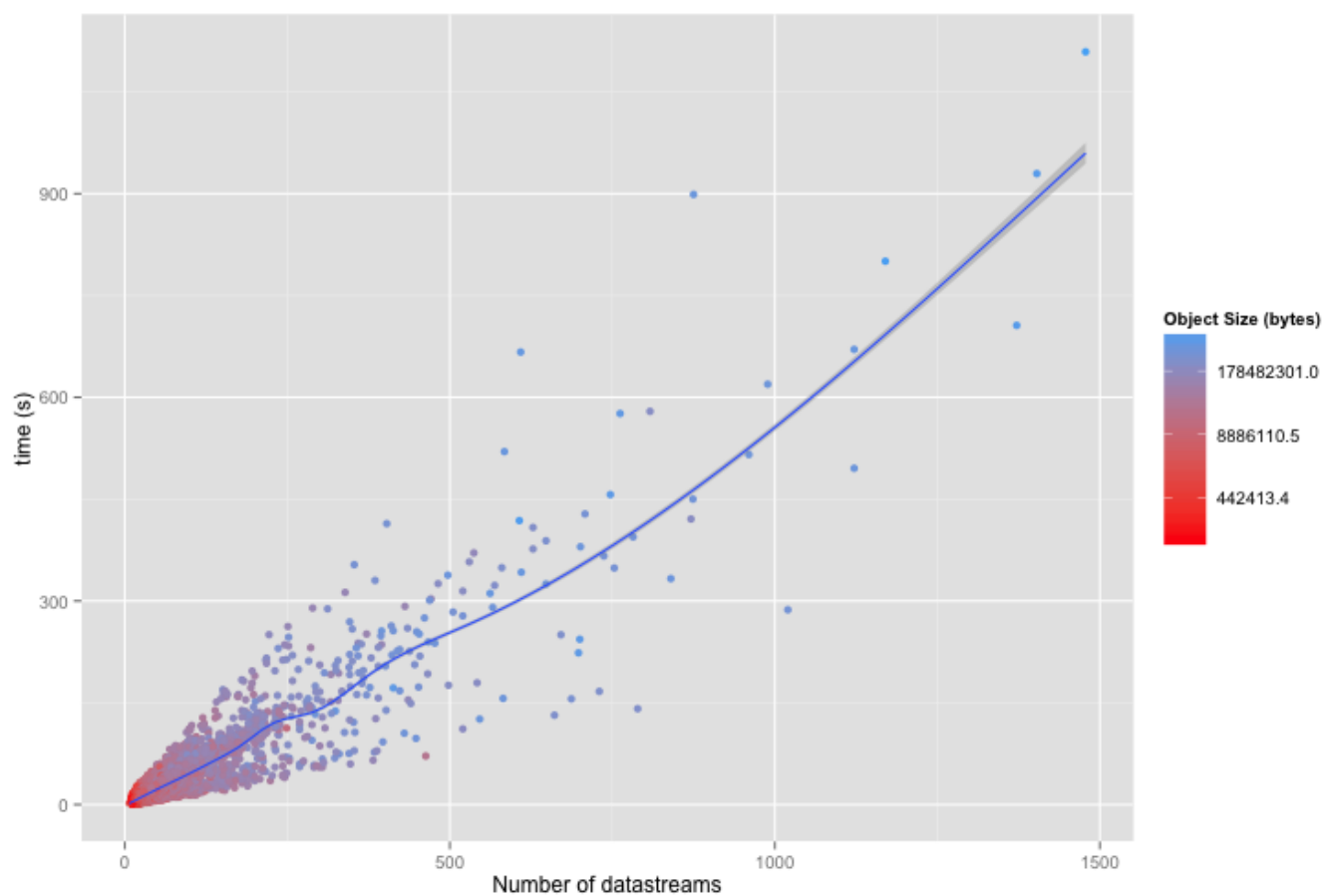




Test 2d: Use a 4-node cluster to do a druid-tree ingest

```
> quantile(create$V4, c(0, .5, .7, .9, .95, .99, 1))  
 0%      50%      70%      90%      95%      99%     100%  
0.9050   9.7360  13.9442  29.6692  48.0332 146.0208 1109.1760
```





Test 3: Realistic Ingest into Fedora 3

Ingest all the data into fcrepo3 making reasonable content modeling assumptions:

- each page as an object
- ?

Using ActiveFedora:

Test 4: Realistic Ingest into Fedora 4

- add RDF as properties on [resources](#)
- Each page as a ordered same-name sibling on an container

Using ldp-client: