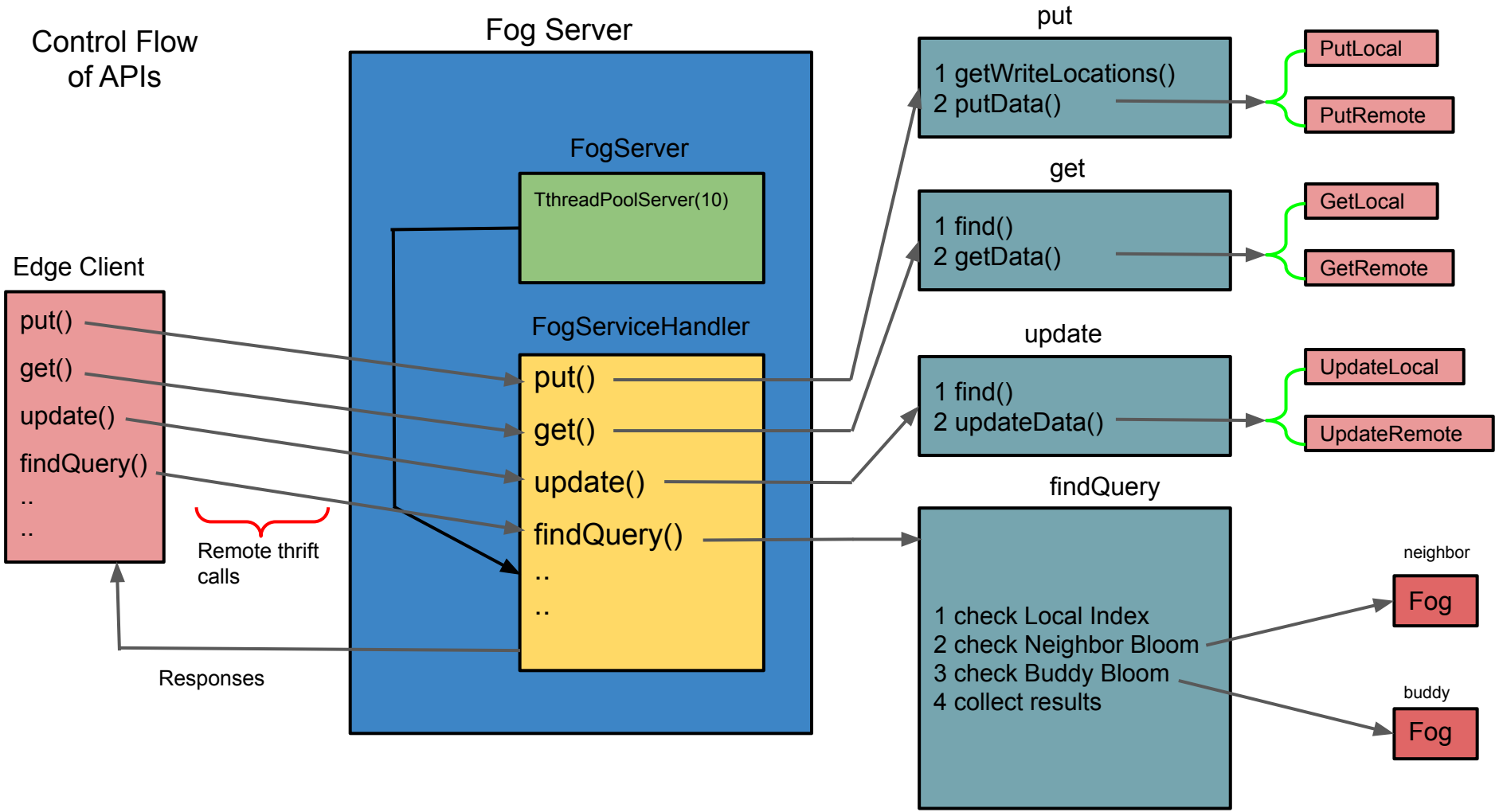
Elfstore Architecture

Sheshadri K R


Control Flow of APIs
Fog Server
FogServer
TthreadPoolServer(10)
FogServiceHandler
put()
get()
update()
findQuery()
..
..
Edge Client
put()
get()
update()
findQuery()
..
..
Remote thrift calls
Responses
put
1 getWriteLocations()
2 putData()
PutLocal
PutRemote
get
1 find()
2 getData()
GetLocal
GetRemote
update
1 find()
2 updateData()
UpdateLocal
UpdateRemote
findQuery
1 check Local Index
2 check Neighbor Bloom
3 check Buddy Bloom
4 collect results
neighbor
Fog
buddy
Fog

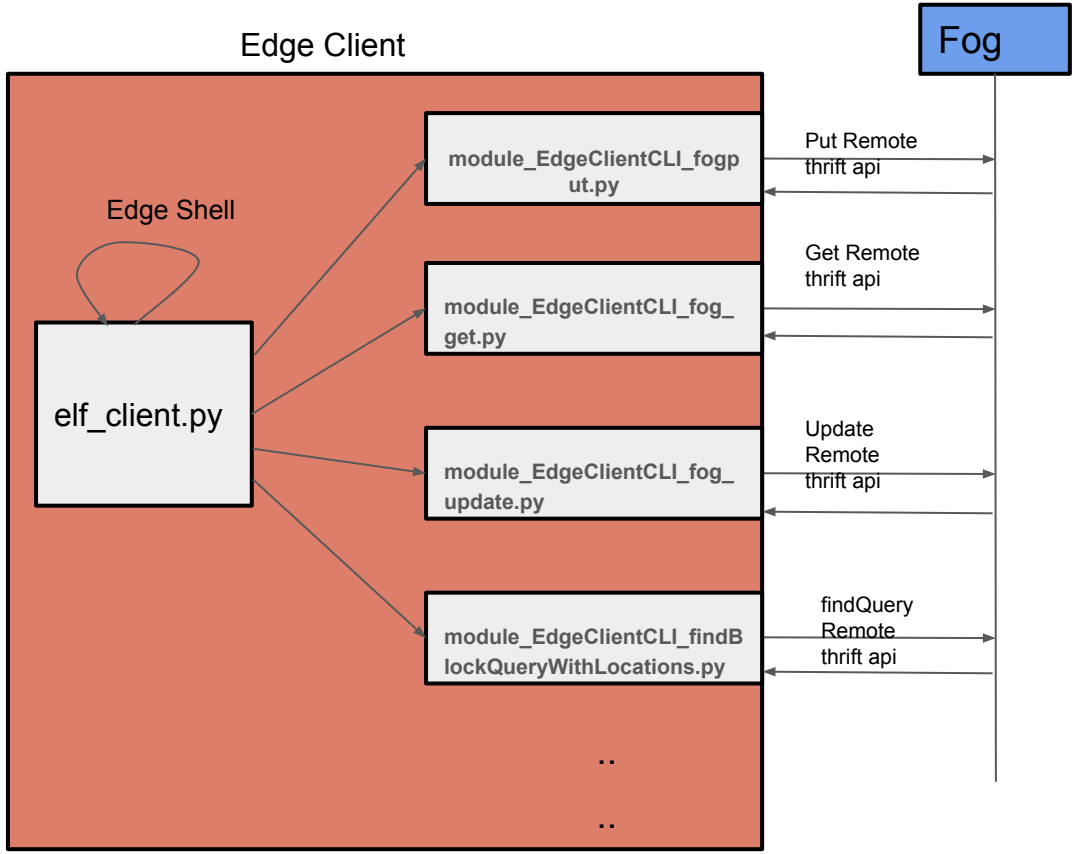
Elfstore - Client

- The client code is written in Python
- It invokes the Server side apis.
- All the callable apis are present in **FogServices.thrift**
- For more information on how to add a Thrift Service Api check the link
  - https://thrift-tutorial.readthedocs.io/en/latest/usage-example.html
- Codebase has a command-line shell
- For examples of using it
  - Refer to **Readme.md**

Elfstore - client

- Major files are listed below
- **Elf_client.py** (main file, starting point)
- Api for each operation is added as a separate python module shown below:
- **module_EdgeClientCLI_fog_put.py**
- **module_EdgeClientCLI_fog_get.py**
- **module_EdgeClientCLI_fog_update.py**
- **module_EdgeClientCLI_findBlockQueryWithLocations.py**

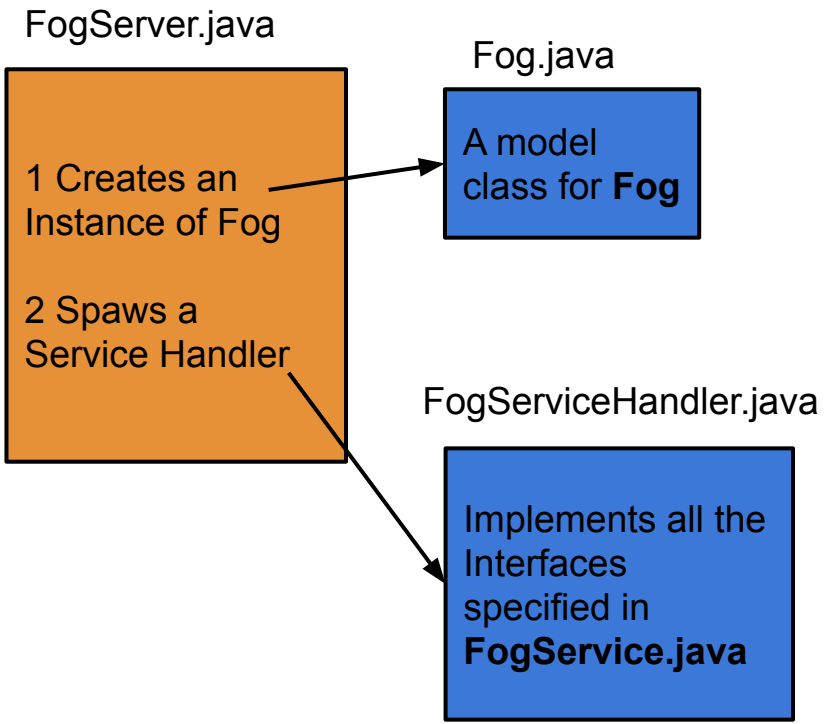
Using the shell commands are listed in file **Commands.txt**



Elfstore - Fog node Server

The main classes in Fog Node Server:

- **FogServer.java**
- **FogServiceHandler.java**
- **Fog.java**



FogServer.java

- **package com.dreamlab.edgefs.controlplane**
- This is the class which runs the Fog Server.
- Fog properties (such as IP, buddies, neighbors) are initialized from here.
- Creates a new **Fog** model instance.
- Creates the Fog, Neighbor and Buddy topology using the configuration file.
  - Cluster.conf (src/main/resources/cluster.conf)
- This class is also responsible for accessing the heartbeat exchange interval and many other properties
  - System.properties (src/main/resources/system.properties)
- Creates a FogServiceHandler instance.

Fog.java

- **package com.dreamlab.edgefs.controlplane**
- This is a model class which holds all the properties of a Fog
  - FogIP, FogID, FogPort, :BuddyID
  - List of all Neighbors, List of all Buddies
- Holds a lot of in-memory index structures such as
  - Block -> Location Map,
  - Metadata -> Block(s) Map
  - Edge -> Status Map
- Holds bloom-filter search index structures of Neighbors and Buddies
  - NeighborInfo (FogExchangeInfo Model class)
  - BuddyInfo (FogExchangeInfo Model class)
- Facilitates exchange of heartbeats between
  - Fog and Neighbor
  - Fog and Buddy
  - Fog and Edge

FogServiceHandler.java

- **package com.dreamlab.edgefs.servicehandler**
- Implements all the interfaces present in **FogService.java**
- It is the invocation point for all the Fog apis such as
  - put()
  - get()
  - update()
  - findBlockQuery()
- Makes use of the **Fog.java** instance to maintain data management.

GlobalReplicaAllocation.java

- **package com.dreamlab.edgefs.controlplane**
- Called during the ***put()*** api invocation
- Invoked by ***getWriteLocations()*** api
- Returns Fog nodes to which blocks will be written
- Is responsible for ***ensuring minimum reliability***.
- Ensures ***min replication*** and ***max replication***
- Chooses the Fog nodes using a ***differential replication scheme*** algorithm

LocalReplicaAllocation.java

- Performs the step of choosing an Edge to write the block
- Invoked by ***identifyLocalReplica()***
- Chooses the edge with a given ***WritePreference***
- ***identifyLocalReplica()*** Ensures that same edge is not chosen twice for a single block

EdgeServer.java

- **package com.dreamlab.edgefs.edge.server**
- This is the starting point for Edge Node.
- Initializes the **Edge** instance which is a model class for Edge Node
- Initializes the **EdgeServiceHandler.java** class which implements the interface **EdgeService.java**

Edge.java

- It is the model class for Edge Node
- **package com.dreamlab.edgefs.edge.model**
- Holds the **EdgeIP, EdgeID, EdgePort** etc.,
- Holds the info of **ParentFog IP, ParentFog port** etc.,
- Has apis to register to Fog
- Has apis to exchange heartbeats with the Parent Fog

EdgeServiceHandler.java

- package **com.dreamlab.edgefs.edge.handler**
- Implements the interface of **EdgeService.java**
- Consists of the apis such as
  - put()
  - get()
  - update()
- It is invoked as a remote thrift api by Fog to perform write/update/get of a block
- Also writes/updates the metadata index related to block

LocalStatsHandler.java

- **Package com.dreamlab.edgefs.misc**
- Considers max, min storage, max, min reliability of current Fog
- Computes the distribution of Edge devices into storage buckets of 4 types
  - HH, HL , LL, LH (Storage, Reliability)
  - Using equi-width and equi-depth partition

GlobalStatsHandler.java

- **Package com.dreamlab.edgefs.misc**
- Considers global max, min storage, max, min reliability of all Fogs
- Computes the distribution of Edge devices into storage buckets of 4 types
  - HH, HL , LL, LH (Storage, Reliability)
  - Using equi-width and equi-depth partition

package com.dreamlab.edgefs.model;

- Contains all the model class for data exchange between Fogs

package com.dreamlab.edgefs.iface;

- Contains all the following interfaces
  - Get Interfaces
  - Put Interfaces
  - Update Interfaces
  - Find Interfaces

package com.dreamlab.edgefs.writes

- Contains all the implementation for write and update operations.

package com.dreamlab.edgefs.reads

- Contains all the implementation for read operations.

package com.dreamlab.edgefs.thrift

- Contains all the thrift related java files, including model classes.
- Interfaces (FogService.java and EdgeService.java)

package com.dreamlab.edgefs.misc

- Contains BloomFilter related classes
- Data exchange format for exchanging bloomfilters between Fogs
- Contains CONSTANTS class which is a final static class.

Configuration and Properties

cluster.conf file

This file defines the Fog overlay, how the Fog nodes form Buddies and neighbors.

Each new line defines a new Fog's IP/Port, ID , buddy-group-ID , and its neighbors

Current Fog IP, current fog port, current fog ID, fog reliability, (buddy-group-ID:neighbor-fog-ID: neighbor-fog-IP:neighbor-fog-Port)

Eg:

127.0.0.1,9090,1,1,0.9,(2:4:127.0.0.1:9093)

127.0.0.1,9091,1,2,0.9,(2:3:127.0.0.1:9092)

127.0.0.1,9092,2,3,0.9,(1:1:127.0.0.1:9090)

127.0.0.1,9093,2,4,0.9,(1:2:127.0.0.1:9091)

1 and 2 are buddies (blue) group 1

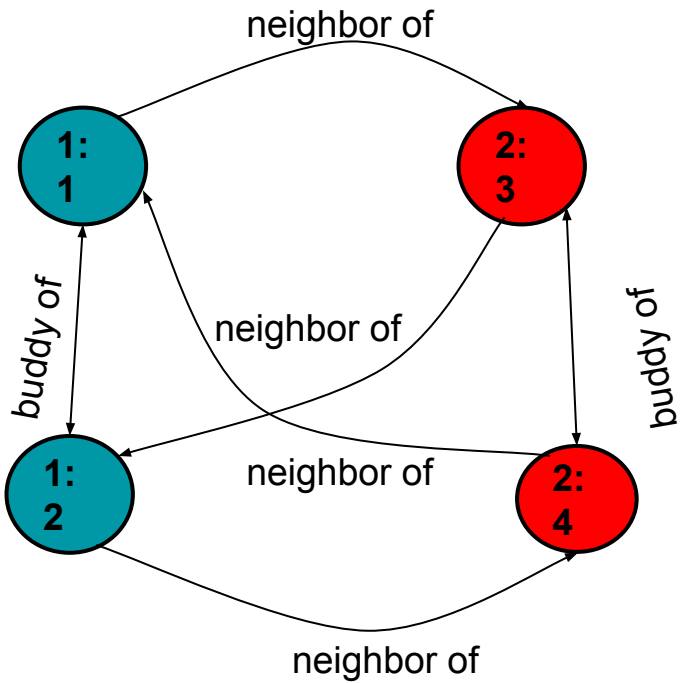
3 and 4 are buddies (red) group 2

4 is a neighbor of 1

3 is a neighbor of 2

1 is a nieghbor of 3

2 is a neighbor of 4



system.properties

Each fog will have the **system.properties** (present in resources) file which decides how frequently will it communicate and exchange crucial data between buddies and neighbors.

The buddy related statistics are explained below

```
#Buddy related properties
#heartbeat interval in seconds
buddy.heartbeat.interval=5
#send intervals when there is a change
buddy.stats.send.heartbeats=6
buddy.bloom.send.heartbeats=12
#forced send intervals even when there is no change
buddy.stats.force.send.heartbeats=18
buddy.bloom.force.send.heartbeats=15
```

Every 5 seconds heartbeats are exchanged between buddies. These are just simple Heartbeat messages which may or may not contain payload

Every 6th heartbeat (which is 5x6 = 30s) send Fog related statistics to a buddy, if there is a change in statistic

Every 6th heartbeat (which is 5x12 = 60s) send bloomfilter index to buddy, if there is a change in statistic

Every 18th heartbeat (which is 18x5 = 90s) send edge related statistics to a buddy whether or not there are any changes

Every 15th heartbeat (which is 15x5 = 75s) send bloomfilter a buddy whether or not there are any insertions into the bloomfilter

Similarly the properties hold for neighbors as well.

How to run?

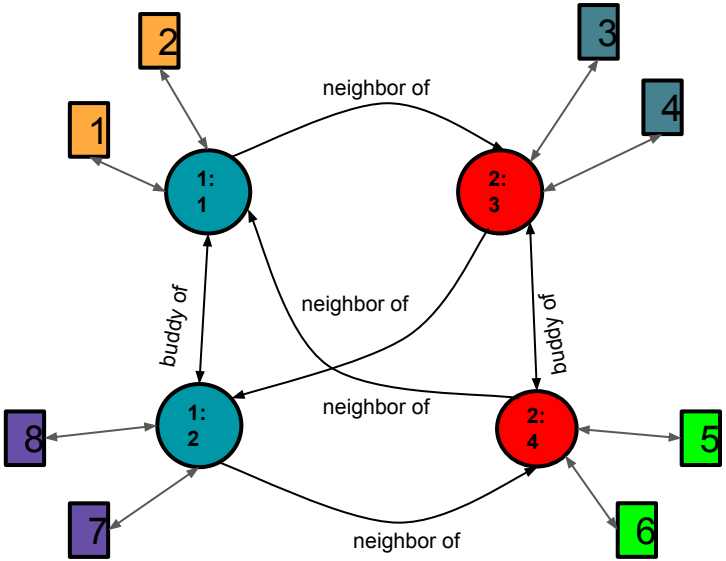
Running in Local Mode

- Configuration of 4 Fogs and 8 Edges
- The Fog IPs and the topology is same as mentioned previously.
- The IP and port of Fog will be picked from **cluster.conf**
- Everytime **cluster.conf** is changed, the jar needs to be recompiled to take changes in the IP and Port of Fogs.

There are 2 phases in running Elfstore

1. Start Fog Server and Edge Server
2. Start the Edge clients

Fogs and Edges will run as processes in your local computer.



Starting Fog and Edge Server

To start the fog:

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.controlplane.FogServer 127.0.0.1 9090 1 1 0.25 0 (Terminal 1)

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.controlplane.FogServer 127.0.0.1 9091 1 2 0.25 0 (Terminal 2)

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.controlplane.FogServer 127.0.0.1 9092 2 3 0.25 0 (Terminal 3)

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.controlplane.FogServer 127.0.0.1 9093 2 4 0.25 0 (Terminal 4)

Explanation of the command above:

java -cp path_to_jar **com.dreamlab.edgefs.controlplane.FogServer** FogIP FogPort BuddyGroupID FogID FogReliability 0

FogIP, FogPort, FogID, Buddy Group ID are self explanatory and follow from **cluster.conf** file.

FogReliability is currently not used so keep 0.25 as the default value

0 is a default parameter, use it as it is.

At the end of this, **4 fog server java processes will be started**. You can verify this by checking htop or ps.

After 10 heartbeats (approximately 1 min) the Fogs will synchronize.

Starting Edge Server

To run EdgeServer, start the file **startedges.sh** (internally it starts 8 Edge Servers which are java processes)

Change the /home/swamiji/edge_logs2/ /home/swamiji/base_logs/ to your local paths where logs are written /~/edge_related_logs/ /~/base_logs/

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.edge.server.EdgeServer 1 127.0.0.1 7000 85 127.0.0.1 9090 /home/swamiji/edge_logs1/ /home/swamiji/base_logs/ &

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.edge.server.EdgeServer 2 127.0.0.1 7001 85 127.0.0.1 9090 /home/swamiji/edge_logs2/ /home/swamiji/base_logs/ &

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.edge.server.EdgeServer 3 127.0.0.1 7002 85 127.0.0.1 9091 /home/swamiji/edge_logs3/ /home/swamiji/base_logs/ &

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.edge.server.EdgeServer 4 127.0.0.1 7003 85 127.0.0.1 9091 /home/swamiji/edge_logs4/ /home/swamiji/base_logs/ &

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.edge.server.EdgeServer 5 127.0.0.1 7004 85 127.0.0.1 9092 /home/swamiji/edge_logs5/ /home/swamiji/base_logs/ &

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.edge.server.EdgeServer 6 127.0.0.1 7005 85 127.0.0.1 9092 /home/swamiji/edge_logs6/ /home/swamiji/base_logs/ &

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.edge.server.EdgeServer 7 127.0.0.1 7006 85 127.0.0.1 9093 /home/swamiji/edge_logs7/ /home/swamiji/base_logs/ &

java -cp target/edgefilesystem-0.1-jar-with-dependencies.jar com.dreamlab.edgefs.edge.server.EdgeServer 8 127.0.0.1 7007 85 127.0.0.1 9093 /home/swamiji/edge_logs8/ /home/swamiji/base_logs/ &

Starting Edge Server

The command to start Edge Server

- java -cp path_to_jar edgeID edgeIP edgePort edgeReliability FogIP FogPort path_to_edge_data path_to_edge_logs

Explanation of the command

- edgeReliability -> An integer whose value lies between 1 to 100. It indicates edge's reliability.
- FogIP,FogPort -> The Fog it needs to contact to get registered.
- Path_to_edge_data -> Path where edge blocks are store(data and meta files)
- Path_to_edge_logs -> Path where edge related logs are written.

The decision of which edge needs to register to which Fog is done **statically**.

In our case, Edge 1&2 registers to Fog1, 3&4 on Fog2 , 5&6 on Fog3 , 7&8 on Fog4.
Once the **startedges.sh** command is executed, 8 edges start and they register to their respective Fog devices (the mapping is given above)

At this point, the edges and fogs are started, keep checking your Fog, until one of the Fog gets the overall picture of number of edges in the system. An output like this is an indication that all Fogs and Edges are started.

****

23:27:40.950 [Thread-7] INFO c.d.edgefs.controlplane.FogServer - edge distribution map is{HH=4, HL=4, LL=0, LH=0}

****

Running in Docker environment using ViOLET

Check this link to find out how dockerised version of Elfstore can be run using ViOLET

https://github.com/ishanSharma07/ElfStoreDocker

Testing the Edge Client

Edge Client

Change to **cli** directory. It is in the same level as that of **src**

cd cli; then execute the following command to start the edge client

python2.7 -W ignore elfs_cli.py edge-config-files/edge1_config.json

```
(base) swamiji@swamiji-HP-Laptop-15-da0xxx:~/eclipse-workspace/ElfStoreBase/cli$ python2.7 -W ignore elfs_cli.py edge-config-files/edge1_config.json
Session Information
Edge ID : 1
Edge IP : 127.0.0.1
Edge Port : 7000
Edge Reliability : 85
Fog IP : 127.0.0.1
Fog Port : 9090
Storage Location : ./DataAndLogs/edge1_data
Client ID : c4ca4238a0b923820dcc509a6f75849b

ElfStore: A Resilient Data Storage Service for Federated Edge and Fog Resources

elfs>
```

This would make edge1 as a client. (like that you'll have 8 edges started)

Registering stream

Try these following commands in the order in the cli shell, in the order, one after the other.

****

Register the stream

regstream --fogIp 127.0.0.1 --fogPort 9090 --id stream_id_test_1 --reli .90 --minReplica 3 --maxReplica 3 --verbose

streamID being registered is **stream_id_test_1**

--reli .90 means the reliability of all the blocks stored in this stream will be at least to be 0.90.

--minReplica 3 means a minimum of 3 copies or replicas of the block will be maintained, for all blocks in this stream.

Even though the reliability requirement is met with only 2 copies, still minimum of 3 copies will be maintained.

--maxReplica 3 means a maximum of 3 copies or replicas of the block will be maintained, for all blocks in this stream.

Even though the reliability requirement is not met with 3 copies, the number of copies will not go beyond 3 copies.

Put blocks

- Put block , block number is 777
- **fogput --path=/home/swamiji/phd/EdgeFS/ElfStore/microbatches/microbatch0.txt --fogIp 127.0.0.1 --fogPort 9090 --streamId=stream_id_test_1 --start=777 --singleBlock --setLease --v**
- {once this command executes you should be able to see "[PUT BLOCK] Parallel puts done" in 1st terminal or Fog-1 }
- Path here indicates the file which will be inserted as block 777
- **--fogIp 127.0.0.1 --fogPort 9090** Indicates the IP and Port of Fog which will handle the put request.
- **--streamId=stream_id_test_1** indicates the stream in which this block will be put, the minReplica, maxReplica and reliabilities are derived from this.
- **-setLease** Indicates lease should be acquired from stream owner before putting block.
- **--start=777** The block number is 777
- **--singleBlock** Indicates that only one block is insterted.
- Note: For multi block insertion, the path should point to a directory Eg: if **/home/swamiji/phd/EdgeFS/ElfStore/microbatches /** has 10 files, and the --start=777 then 10 blocks will be inserted starting from 777 through 786.

Get blocks

- **fogget --start=777 --v**
- {once this command executes you should be able to see "[DS256] The read is local" in 1st terminal }
- If no ip or port is specified, a default fop ip,port which is statically assigned to the current Edge is used. This mapping is available at **/cli/edge-config-files/edge1_config.json**
- Else specific fog-ip port can be mentioned using **--fogIp and --fogPort**.
- Similar to put, multiple blocks can be read.

Other commands

This command is similar to put

fogupdate --path=/~/EdgeFS/ElfStore/microbatches/sample.txt --fogIp 127.0.0.1 --fogPort 9093 --streamId=stream_id_test_1 --start=777 --singleBlock --setLease --v

Reading the metadata files of blocks for the given keylist

getmeta --fogIp 127.0.0.1 --fogPort 9090 --edgeId 1 --edgeIp 127.0.0.1 --edgePort 7001 --mbid 11 --keylist cds

--mbid 11 The block being read, **--keylist cds** The key-list for which metadata values need to be read.

--fogIp 127.0.0.1 --fogPort 9090 IP and Port of Fog being contacted

--edgeIp 127.0.0.1 --edgePort 7001 The edge in which the block 11 is present

Other commands

Find query, the query parameters are present in the file module_EdgeClientCLI_findBlockQueryWithLocations.py

findwithloc --fogIp 127.0.0.1 --fogPort 9090 --matchpref 1 --findloccount 0

--findloccount 0 (returns no location) 1 (returns single location) 2 (returns all locations)

-matchpref 1 indicates OR comparison, 2 indicates AND comparison of the query parameters

--fogIp 127.0.0.1 --fogPort 9090 IP and Port of Fog being contacted

Adding new Thrift Services

Services in ElfStore

FogServices.thrift

EdgeServices.thrift

Whenever a new service needs to be added in **FogServices.thrift**, which will reflect in a new service in **FogServiceHandler.java**

Add the following service in the in **FogServices.thrift** file,

byte sampleService(1: i64 arg1,2:string arg2, 3: NodeInfoData arg3);

Then run

thrift --gen java FogServices.thrift

thrift --gen py FogServices.thrift

Two files will be generated

gen-java and **gen-py at the** FogServices.thrift folder level.

Copy contents inside **/gen-java/com/dreamlab/edgefs/thrift/** to **/src/main/java/com/dreamlab/edgefs/thrift/ (Add change to server code)**

Copy contents of **gen-py to /cli/gen-py (This would add the change to the client code)**

Now, implement this service in FogServiceHandler.java.

Recompile using "mvn clean compile assembly:single"

After this from the client side, the new service should be accessible to invoke with **sampleService() just like any other previous services.**

You can add is as a new module_file in the Client side code and make it accessible as a client side command like fogput or fogget.