Google File System(GFS) - user-level application

requirement: performance, scalability, reliability, availability

situations: 1) common component failure 2) huge files 3) appending is common rather than overwrite 4)concurrency requirement 5)

REASON For multiple appends

FAILURES -> IF one node IS SUCCESS, AND THE ANOTHER FAILS, WHEN RETRING,THE SUCCESSFUL ONE WILL HAVETWICE replicas.

## design overview

interface: create, delete, open, close, read, write, **snapshot, record append**

one master, multiple chunkservers, accessed by multiple clients

**master**: all file system metadata as well as system-wide activities, a client asks the master for the chunkserver it should contact.

metadata: file + chunk namespaces, mapping from files to chunks, locations of replicas(ask the information at startup because there are too many changes due to failures, its hard to keep it persistent)

having the \*\*operation log\*\* for first two, and it is replicated on multiple remote machines, only reply to the clients when the operation logs are flushed to disks.

**chunkserver**: no cache(the bottom linux os file system may have cache)

**client**: no cache

large chunk size of 64MB ->

* reduce the interaction between client and master
* the chunk number can be reduced that the network overhead reduced
* reduce size of metadata in the server
* (drawback) the one chunkserver may become hot spots that many clients visit -> store more replicas

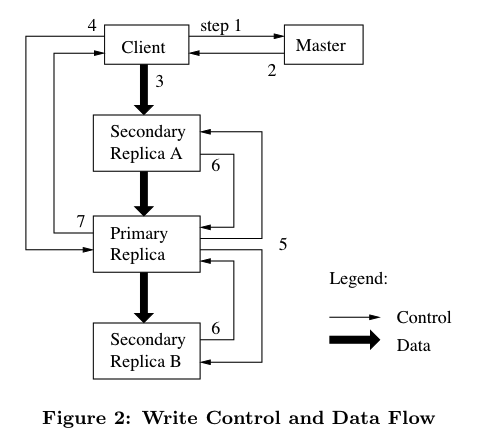
file state:

* consistent -> all the clients will see the same data from different replicas
* defined -> consistent and all the mutations are done

## system interactions

all the chunk replicas -> one primary replica -> assign with a chunk lease

primary will pick a serial order of mutations



snapshot: implemented in copy-on-write

record append: if the file size is larger than the current lasting trunk, just append it to the next chunk. maybe there is garbage generated from this procedure. but the mutation is atomic.

it is consistent among replicas of one trunk, but may not defined among different trunks of one request of different clients

## master operations

locking: path-along checking…