1. Design
   1.1. Algorithm and Replication

   We put our nodes and files into a **virtual ring** as a DHT. The IP address of each node is used to calculate its position in the ring. The remote filename is used to calculate its primary host. The (two) additional replicas are at the following two hosts in the ring via **Active Replication**. To simplify our design and hence implementation, we **avoid master** node of the whole system and hence Leader Election. Instead, every node has a full view of the file table of this DHT, and is able to handle requests and failures straightforward. So, the metadata is safe after whichever node fails, and requests needn't be routed first to master, whose failure will change the entry point for later requests. We also **eliminate client** role because it's a trivial relay but entails entry point issues. The server itself is used as a client instead (directories are distinguished). With this design, PUT and GET can be easily implemented by **RPC** after looking up the local file table on the server. When failures happen, the alive successor will help handle the re-replication and update file table for each alive node, still by RPC. To deal with two simultaneous failures, when a single failure is detected, the gossiper will **keep checking** for a safe time interval before reporting to server in case there's another failure and those failures should be **handled in a batch**.

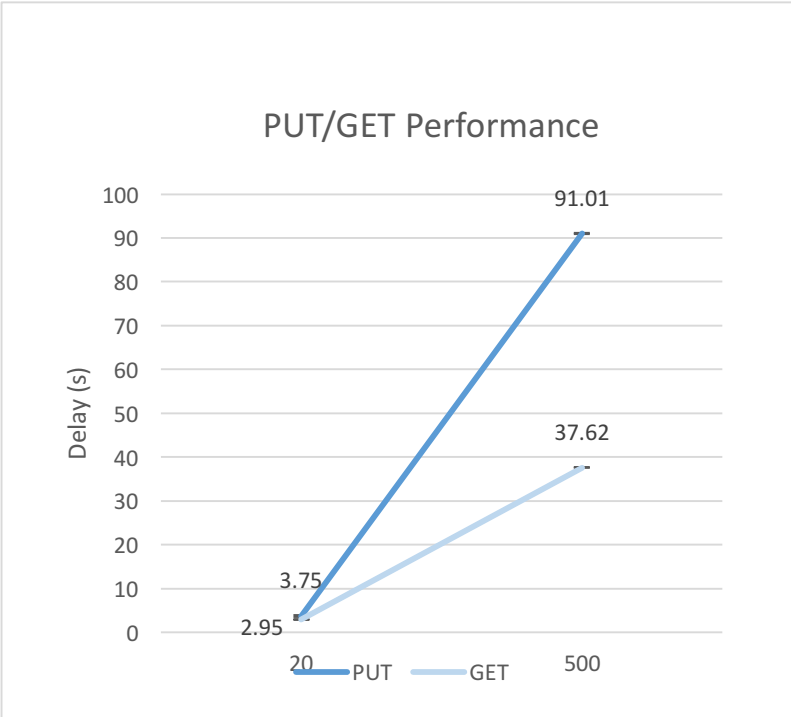
   1.2. Integration of Former MP's

   Both MP1 and MP2 are helpful in MP3. MP1 is used to debug, and MP2 is used to maintain the metadata and handle re-replication.

2. Performance

Test case		**Average delay**	**Standard deviation**
Re-Replication (20MB)		1.49s	1.08s
Master Election		N/A (no master)	N/A (no master)
PUT	20MB	3.75s	0.17s
	500MB	91.01s	0.14s
GET	20MB	2.95s	0.02s
	500MB	37.62	0.04s
PUT	11G	2214.86s	14.21s

   The PUT of 11G file is quite slow; it may be optimized if we use FTP to transfer the file.



The delay's proportional trend against the file size is reasonable because the time is mostly used to transfer the file. It's also noticed that the standard deviations are quite small, it's because that the file transfer performance is very stable in the VM cluster. It's not perfectly proportional, mainly because of the overhead of other computation.