

gDocs测试文档

我们进行软件测试的总体思路为先单元测试、集成测试，再到系统测试，端到端测试。对于分布式文件系统，我们主要是用go_test框架进行了集成测试和系统测试，而后使用Postman人工操作进行复杂操作下的Bug筛查（类似于模糊测试）。对于gDoc应用，我们主要使用Postman和人工操作进行端到端测试。

此测试文档总结了在开发GFS时用来验证功能和功能性需求的测试用例。

分布式文件系统测试

测试用例

我们为**awesomeGFS**（即我们项目的DFS）设计了一共**21**个测试用例，覆盖了对Master、ChunkServer、Client等对象的单元测试以及一致性、容错测试等。所有测试都在**awesomeGFS/awesome_test.go**文件中。具体每个测试用例的含义如下：

- 基础文件操作测试（共9个测试）

```
44  /*
45  *  TEST SUITE 1 - Basic File Operation
46  */
47  func TestUtil(t *testing.T) {...}
74
75  func TestCreateFile(t *testing.T) {...}
103
104  func TestMkdirRecursive(t *testing.T) {...}
134
135  func renameTest(sPath, tPath string, t *testing.T) {...}
205
206  func TestRenameFile(t *testing.T) {...}
221
222  func TestMkdirDeleteList(t *testing.T) {...}
300
301  func TestRPCGetChunkHandle(t *testing.T) {...}
326
327  func TestWriteChunk(t *testing.T) {...}
340
341  func TestReadChunk(t *testing.T) {...}
361
362  // check if the content of replicas are the same, returns the number of replicas
363  func checkReplicas(handle gfs.ChunkId, length int, t *testing.T) int {...}
393
394  func TestReplicaEquality(t *testing.T) {...}
405
406  func TestAppendChunk(t *testing.T) {...}
```

- **TestUtil**：测试了util.go文件中的一些util函数，包括SplitFilePath、FindCommonPath等对路径进行处理的util函数。
- **TestCreateFile**：主要对Master的Create API进行测试，测试了其正确性以及对错误参数的处理，包括重复创建同名文件，在不存在的路径下创建文件，以及在文件下创建文件三种错误文件创建场景。

- **TestMkdirRecursive**: 测试了Master的Mkdir API，即递归创建文件夹的接口的正确性，检查其是否能递归的创建新文件夹。
- **TestRenameFile**: 测试了Master的Rename API的多种场景。
 - 测试场景：包括source和target在同一个文件夹下、source和target在父子文件夹下、source和target在不同路径（在不同文件夹下且两者的目录无父子关系）下等等。
 - 测试用例设计初衷：Master的Rename API是Master的namespace_manager下最复杂的一个API，因为namespace_manager对于namespace的并发写问题采用的方案是细粒度锁，也就是为每个dir都加一把读写锁，而不是使用一把大锁锁住整个namespace。因此Rename的实现中包括了对source路径和target路径的共同路径解析，对于source和target共同路径下的dir只需要拿一把读锁，因此不能直接对source和target路径调用lockParents和unlockParents（否则会重复拿锁），而拆解剩余的source和target的非共同路径部分可以再使用lockParents和unlockParents来进行锁处理。此外需要注意，不同于其他路径上的目录，source文件和target文件的根目录需要拿写锁，这也增加了Rename函数的复杂性。
- **TestMkdirDeleteList**: 对Master的Mkdir、Create、List和Delete API进行综合性测试。首先调用Mkdir在根目录下创建文件夹（并测试重复创建），而后调用Create在根目录以及新创建的文件夹下创建若干文件（并测试重复创建文件），调用List列出几个文件夹下的文件，检查List是否能覆盖所有子文件，最后调用Delete删除文件夹下文件后再次调用List检查删除是否有效。
- **TestRPCGetChunkHandle**: 主要测试Master的GetChunkHandle API。首先调用Create在根目录下创建新文件，并使用新文件的路径调用Master的GetChunkHandle接口两次，检查两次结果是否相同。最后对错误参数处理进行测试，调用GetChunkHandle接口时传入参数Index为2（即找到该文件的第三个Chunk，由于该文件刚被创建，不应该有第三个Chunk），检查是否能够返回错误。
- **TestWriteChunk**: 对Client的WriteChunk API进行正确性测试。
- **TestReadChunk**: 对Client的ReadChunk API进行正确性测试。
- **TestReplicaEquality**: 对Chunkserver上保存的Chunk做一致性检查，检查对同一块Chunk做的Replication是否相同。

● Client API测试（共2个测试）

```

409  /*
410   * TEST SUITE 2 - Client API
411   */
412
413  // if the append would cause the chunk to exceed the maximum size
414  // this chunk should be pad and the data should be appended to the next chunk
415  func TestPadOver(t *testing.T) {...}
416
417  // big data that invokes several chunks
418  func TestWriteReadBigData(t *testing.T) {...}
419
420

```

- **TestPadOver**: 对Client的Append API进行了一个特殊场景测试。即当Append的内容会溢出文件原有的最后一个Chunk时，应当将溢出部分的内容写入新的Chunk当中，新Chunk将取代原有Chunk成为新的最后一个Chunk。该用例的测试方法为，首先新建文件，对其进行4次常规的Append写入（即写入同一个Chunk，写入直到该Chunk只剩下最后4个byte），而后进行第五次Append写入，即特殊场景写入（写入5个byte，溢出原有的最后一个Chunk），检查返回值是否正常。
- **TestWriteReadBigData**: 用较大的数据对Client的Read、Write API进行测试。具体测试方法为，首先创建新文件，构造大小为3个Chunk size的buffer并调用Write API写入文件；然后调用Read API读取3个Chunk size大小的文件数据（从offset=0开始），并比对是否与之前写入的数据一致；最后进行特殊场景的测试，尝试调用Read API读取三个Chunk size大小的文件数据（offset为文件中央，也就是1.5个Chunk size的位置），并检查是否有报错，因为此时文件已经读到结尾应该返回EOF。

● 并发控制读写锁测试（共1个测试）

```

490
491 // TestWriteReadLock is used to test the correctness of concurrency control implemented by Zookeeper.
492 // the result show below:
493 // awesome_test.go:408: test start:          2021-07-12 10:59:17.386257 +0800 CST m=+0.304993204
494 // awesome_test.go:412: first write end:      2021-07-12 10:59:19.425276 +0800 CST m=+2.344035573
495 // awesome_test.go:417: second write end:     2021-07-12 10:59:24.441531 +0800 CST m=+7.360346829
496 // awesome_test.go:427: second read end:      2021-07-12 10:59:26.450982 +0800 CST m=+9.369820562
497 // awesome_test.go:422: first read end:       2021-07-12 10:59:29.457872 +0800 CST m=+12.376744804
498 // awesome_test.go:432: third write end:      2021-07-12 10:59:31.4747 +0800 CST m=+14.393595999
499 // this read write lock is fair. It doesn't prefer reader or writer. And achieve that: Read and write operations are
500 // mutually exclusive and read operations can execute concurrently.
501 func TestWriteReadLock(t *testing.T) {...}
540

```

- **TestWriteReadLock**: 对使用Zookeeper的分布式读写锁的算法逻辑进行了验证，由于文件系统读写时间难以确定，我们使用相同的读写锁函数，编写了虚假的读写过程来控制读写时间，模拟读写争用的问题，从log中可以发现，读者写者拿锁放锁的顺序与预期完全一致。从而证明了读写锁算法的有效性。（为了设计该情景，我们的测试代码较长，每个代码块均包含注释）

● 链式主从备份测试（共3个测试）

```

540
541 // TestListenMasterAndBackup is used to test the correctness of "master slave backup in chain form",
542 // which need everyone in chain know their master and their backup at any time. We implement this
543 // work in Zookeeper. The test below is the situation that ---
544 // there are 4 masters in chain, and their dead order is: master 2 dead, master 1 dead, master 4 dead, and then master 3 dead.
545 // Three possibilities of chain collapse are constructed:
546 // (1) the head dead
547 // (2) the middle node dead
548 // (3) the tail dead
549 // watch the log in test result, we can see correct reflect of live nodes in every possibilities.
550 func TestListenMasterAndBackup(t *testing.T) {...}
593
594 // TestMasterBackupWork is used to test the correctness of "chain master slave": two real master nodes form "chain
595 // master slave backup", and when the master collapse, the slave works immediately, which means gfs clients can find
596 // slave master node and slave master node hold lasted metadata.
597 func TestMasterBackupWork(t *testing.T) {...}
659
660 // TestChunkServerWatchAndOtherMetaInBackupMaster is used to test the correctness of chunkServer's watching at primary
661 // master address and backup master can maintain all the metadata of 'chunk manager' and 'chunk server manager' correctly.
662 // So when the primary master is dead, the backup master can play role like "tall clients where the chunk store" or
663 // "who is primary" or "which chunk need reReplica" and so on.
664 func TestChunkServerWatchAndOtherMetaInBackupMaster(t *testing.T) {...}
747

```

- **TestListenMasterAndBackup**: 验证链式主从备份中主节点能否正确监听链上节点状态的变化，时刻维护自己的“主Master”和“从Master”，设计的链共包含5个Master节点，制造了三种崩溃情况（头节点崩溃，中间节点崩溃，尾部节点崩溃），我们证明3种事件都产生了正确的结果。
- **TestMasterBackupWork**: 验证链式主从备份能够起到容错的作用，此时编写完了关于命名空间的元数据备份逻辑，在Master间互相确定备份关系无误的情况下，检测真的发生崩溃后从节点是否成功保存了重要的元数据，并提供服务。测试虽然只测了Create API，但通过Postman我们证明其他命名空间的操作的正确性。
- **TestChunkServerWatchAndOtherMetaInBackupMaster**: 验证所有元数据都可正确备份。此时我们完成了ChunkServer对提供服务的Master的监听，进一步实现了Master备份节点对所有元数据的备份，使之在“主Master”崩溃时，能立即接管对ChunkServer的管理，不会出现不认识原有ChunkServer，版本号错误等问题，至此所有API均在Master前后崩溃的条件下完成了测试。

● 动态添加节点与负载均衡测试（共1个测试）

```

747
748 //TestAddChunkServerDynamicAndLoadBalance is used to check add ChunkServer dynamically and load balance.
749 func TestAddChunkServerDynamicAndLoadBalance(t *testing.T) {...}
796

```

- **TestAddChunkServerDynamicAndLoadBalance**: 验证负载均衡算法的效果，Chunk迁移到新加入的节点，以及用写文件再读出来的方式验证Chunk迁移的正确性。

● 容错测试（共5个测试）

```

796  /*
797  * TEST SUITE 3 - Fault Tolerance
798  */
799
800  // Shutdown all servers in turns. You should perform re-replication well
801  func TestReReplication(t *testing.T) {...}
848
849  // Shut master down after a few actions, and restart it to see if the log works
850  func TestMasterLog(t *testing.T) {...}
924
925  // after a few actions, do checkpoint for master, then restart it to see if the checkpoint works
926  func TestMasterCheckPoint(t *testing.T) {...}
998
999  // Shut chunkServer down after a few actions, and restart it to see if the log works
1000 func TestChunkServerLog(t *testing.T) {...}
1060
1061 // after a few actions, do checkpoint for chunkServer, then restart it to see if the checkpoint works
1062 func TestChunkServerCheckPoint(t *testing.T) {...}

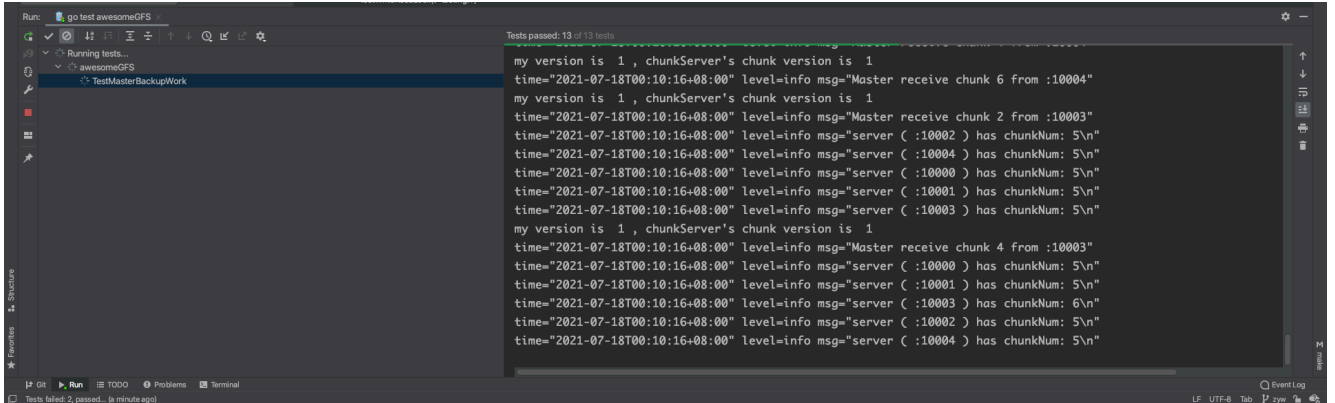
```

- **TestReReplication**：测试系统在若干chunkserver崩溃后是否能保持对某个chunk的Replication。
 - 测试方法：创建一个新chunk后，使用Shutdown()让若干台chunkserver宕机（该函数的作用为屏蔽对该chunkserver的所有RPC请求，使得该chunkserver对外界RPC不做任何反应，从外界看来该chunkserver可以视作死亡）再逐个恢复，等到所有chunkserver恢复后再等候一段时间，最后向master发送请求查询该chunk对应的chunkserver，若数量高于一定值则视为保持了Replication，测试通过。
 - 对于这种容错场景，awesomeGFS系统的应对措施为：一方面，通过心跳机制检查chunkserver是否存活，若检查到有chunkserver死亡，则将其储存的chunk信息进行更新（chunk到chunkserver的mapping要去掉这个死掉的chunkserver）；另一方面，如果有chunk的replica数量过低，则添加至master管理的replicasNeedList中，每隔一段时间会对这个list中的chunk做ReReplication，以保证每个chunk的Replication数量足够。
- **TestMasterLog**：测试单点Master在宕机后是否能从Log当中恢复出之前的数据，也就是Log的正确性。由于CheckPoint也已经在Master代码中实现，测试的时候需要保证在Master尚未做CheckPoint时就进行崩溃处理。具体测试过程为，首先清空log文件，启动一个Master并制造一些数据（通过Mkdir和Create），检查Master在崩溃前确实写入了这些数据（通过List），而后在Master做CheckPoint之前对Master进行崩溃处理（调用Shutdown()函数），检查Master在崩溃后恢复了这些数据（通过List），由于此时没有Checkpoint，因此如果Master在崩溃后恢复了数据就可以证明Master Log的正确性，最后将该Master关闭，并清空log文件。
- **TestMasterCheckPoint**：测试单点Master在宕机后是否能从CheckPoint当中恢复出之前的数据，也就是CheckPoint的正确性。由于Log也已经在Master代码中实现，测试的时候需要保证剔除Log的影响。具体测试过程为，首先清空log文件，启动一个Master并制造一些数据（通过Mkdir和Create），检查Master在崩溃前确实写入了这些数据（通过List），而后强制Master做CheckPoint（若不做强制CheckPoint则数据都在Log当中）并立刻对Master进行崩溃处理（调用Shutdown()函数），检查Master在崩溃后恢复了这些数据（通过List），由于此时没有Log的影响，因此如果Master在崩溃后恢复了数据就可以证明Master CheckPoint的正确性，最后将该Master关闭，并清空log文件。
- **TestChunkServerLog**：测试单点ChunkServer在宕机后是否能从Log当中恢复出之前的数据，也就是Log的正确性。具体测试过程和TestMasterLog类似。
- **TestChunkServerCheckPoint**：测试单点ChunkServer在宕机后是否能从CheckPoint当中恢复出之前的数据，也就是CheckPoint的正确性。具体测试过程和TestChunkServerLog类似。

测试结果

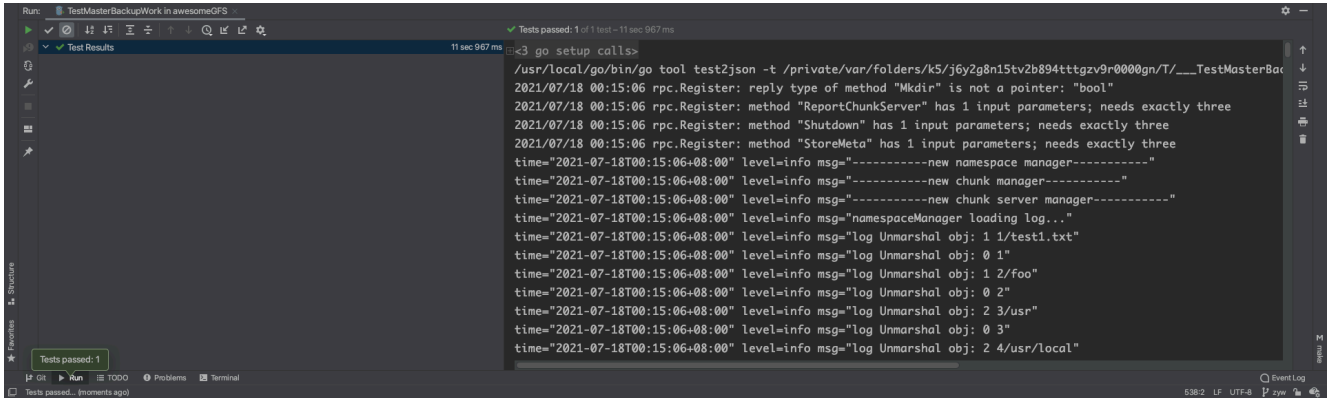
注意：每次测试前需要清空gfs/log路径下的所有log文件

- 前13个测试可以一次性连续通过，结果如下图所示：

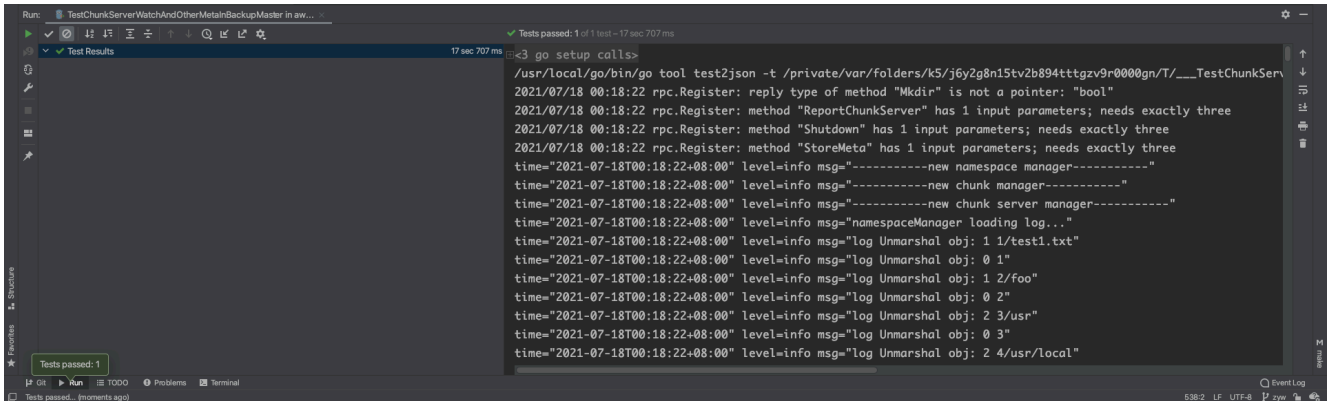


```
Run: go test awesomeGFS
Tests passed: 13 of 13 tests
my version is 1, chunkServer's chunk version is 1
time="2021-07-18T00:10:16+08:00" level=info msg="Master receive chunk 6 from :10004"
my version is 1, chunkServer's chunk version is 1
time="2021-07-18T00:10:16+08:00" level=info msg="Master receive chunk 2 from :10003"
time="2021-07-18T00:10:16+08:00" level=info msg="server ( :10002 ) has chunkNum: 5\n"
time="2021-07-18T00:10:16+08:00" level=info msg="server ( :10004 ) has chunkNum: 5\n"
time="2021-07-18T00:10:16+08:00" level=info msg="server ( :10000 ) has chunkNum: 5\n"
time="2021-07-18T00:10:16+08:00" level=info msg="server ( :10001 ) has chunkNum: 5\n"
time="2021-07-18T00:10:16+08:00" level=info msg="server ( :10003 ) has chunkNum: 5\n"
my version is 1, chunkServer's chunk version is 1
time="2021-07-18T00:10:16+08:00" level=info msg="Master receive chunk 4 from :10003"
time="2021-07-18T00:10:16+08:00" level=info msg="server ( :10000 ) has chunkNum: 5\n"
time="2021-07-18T00:10:16+08:00" level=info msg="server ( :10001 ) has chunkNum: 5\n"
time="2021-07-18T00:10:16+08:00" level=info msg="server ( :10003 ) has chunkNum: 6\n"
time="2021-07-18T00:10:16+08:00" level=info msg="server ( :10002 ) has chunkNum: 5\n"
time="2021-07-18T00:10:16+08:00" level=info msg="server ( :10004 ) has chunkNum: 5\n"
```

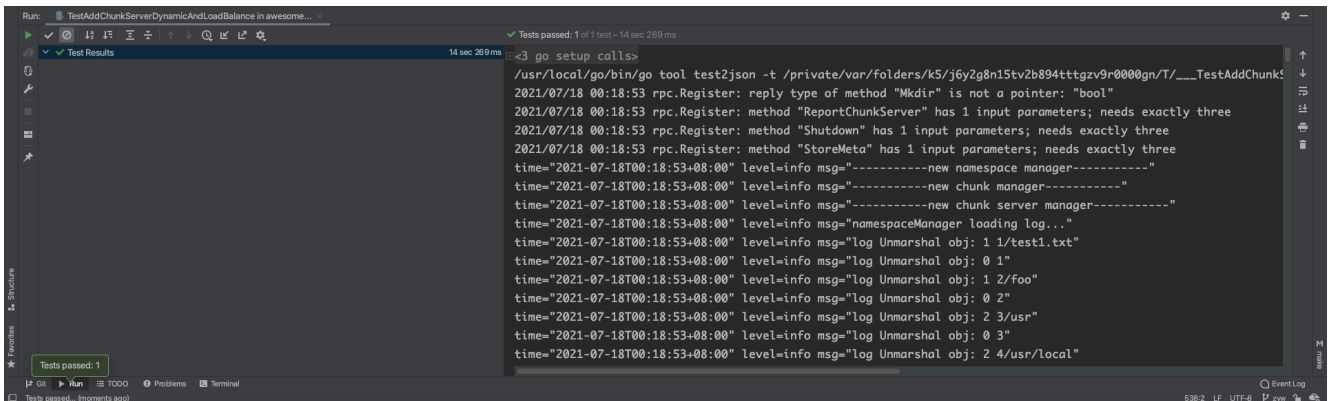
- 由于测试用例之间存在间接影响，后8个测试用例需要单独跑，测试结果按次序如下图所示：



```
Run: TestMasterBackupWork in awesomeGFS
Tests passed: 1 of 1 test - 11 sec 967 ms
<3 go setup calls>
/usr/local/go/bin/go tool test2json -t /private/var/folders/k5/j6y2g8n15tv2b894tttgzv9r0000gn/T/___TestMasterBackupWork
2021/07/18 00:15:06 rpc.Register: reply type of method "Mkdir" is not a pointer: "bool"
2021/07/18 00:15:06 rpc.Register: method "ReportChunkServer" has 1 input parameters; needs exactly three
2021/07/18 00:15:06 rpc.Register: method "Shutdown" has 1 input parameters; needs exactly three
2021/07/18 00:15:06 rpc.Register: method "StoreMeta" has 1 input parameters; needs exactly three
time="2021-07-18T00:15:06+08:00" level=info msg="-----new namespace manager-----"
time="2021-07-18T00:15:06+08:00" level=info msg="-----new chunk manager-----"
time="2021-07-18T00:15:06+08:00" level=info msg="-----new chunk server manager-----"
time="2021-07-18T00:15:06+08:00" level=info msg="namespaceManager loading log..."
time="2021-07-18T00:15:06+08:00" level=info msg="log Unmarshal obj: 1 1/test1.txt"
time="2021-07-18T00:15:06+08:00" level=info msg="log Unmarshal obj: 0 1"
time="2021-07-18T00:15:06+08:00" level=info msg="log Unmarshal obj: 1 2/foo"
time="2021-07-18T00:15:06+08:00" level=info msg="log Unmarshal obj: 0 2"
time="2021-07-18T00:15:06+08:00" level=info msg="log Unmarshal obj: 2 3/usr"
time="2021-07-18T00:15:06+08:00" level=info msg="log Unmarshal obj: 0 3"
time="2021-07-18T00:15:06+08:00" level=info msg="log Unmarshal obj: 2 4/usr/local"
```



```
Run: TestChunkServerWatchAndOtherMetaInBackupMaster in awesomeGFS
Tests passed: 1 of 1 test - 17 sec 707 ms
<3 go setup calls>
/usr/local/go/bin/go tool test2json -t /private/var/folders/k5/j6y2g8n15tv2b894tttgzv9r0000gn/T/___TestChunkServerWatchAndOtherMetaInBackupMaster
2021/07/18 00:18:22 rpc.Register: reply type of method "Mkdir" is not a pointer: "bool"
2021/07/18 00:18:22 rpc.Register: method "ReportChunkServer" has 1 input parameters; needs exactly three
2021/07/18 00:18:22 rpc.Register: method "Shutdown" has 1 input parameters; needs exactly three
2021/07/18 00:18:22 rpc.Register: method "StoreMeta" has 1 input parameters; needs exactly three
time="2021-07-18T00:18:22+08:00" level=info msg="-----new namespace manager-----"
time="2021-07-18T00:18:22+08:00" level=info msg="-----new chunk manager-----"
time="2021-07-18T00:18:22+08:00" level=info msg="-----new chunk server manager-----"
time="2021-07-18T00:18:22+08:00" level=info msg="namespaceManager loading log..."
time="2021-07-18T00:18:22+08:00" level=info msg="log Unmarshal obj: 1 1/test1.txt"
time="2021-07-18T00:18:22+08:00" level=info msg="log Unmarshal obj: 0 1"
time="2021-07-18T00:18:22+08:00" level=info msg="log Unmarshal obj: 1 2/foo"
time="2021-07-18T00:18:22+08:00" level=info msg="log Unmarshal obj: 0 2"
time="2021-07-18T00:18:22+08:00" level=info msg="log Unmarshal obj: 2 3/usr"
time="2021-07-18T00:18:22+08:00" level=info msg="log Unmarshal obj: 0 3"
time="2021-07-18T00:18:22+08:00" level=info msg="log Unmarshal obj: 2 4/usr/local"
```



```
Run: TestAddChunkServerDynamicAndLoadBalance in awesomeGFS
Tests passed: 1 of 1 test - 14 sec 269 ms
<3 go setup calls>
/usr/local/go/bin/go tool test2json -t /private/var/folders/k5/j6y2g8n15tv2b894tttgzv9r0000gn/T/___TestAddChunkServerDynamicAndLoadBalance
2021/07/18 00:18:53 rpc.Register: reply type of method "Mkdir" is not a pointer: "bool"
2021/07/18 00:18:53 rpc.Register: method "ReportChunkServer" has 1 input parameters; needs exactly three
2021/07/18 00:18:53 rpc.Register: method "Shutdown" has 1 input parameters; needs exactly three
2021/07/18 00:18:53 rpc.Register: method "StoreMeta" has 1 input parameters; needs exactly three
time="2021-07-18T00:18:53+08:00" level=info msg="-----new namespace manager-----"
time="2021-07-18T00:18:53+08:00" level=info msg="-----new chunk manager-----"
time="2021-07-18T00:18:53+08:00" level=info msg="-----new chunk server manager-----"
time="2021-07-18T00:18:53+08:00" level=info msg="namespaceManager loading log..."
time="2021-07-18T00:18:53+08:00" level=info msg="log Unmarshal obj: 1 1/test1.txt"
time="2021-07-18T00:18:53+08:00" level=info msg="log Unmarshal obj: 0 1"
time="2021-07-18T00:18:53+08:00" level=info msg="log Unmarshal obj: 1 2/foo"
time="2021-07-18T00:18:53+08:00" level=info msg="log Unmarshal obj: 0 2"
time="2021-07-18T00:18:53+08:00" level=info msg="log Unmarshal obj: 2 3/usr"
time="2021-07-18T00:18:53+08:00" level=info msg="log Unmarshal obj: 0 3"
time="2021-07-18T00:18:53+08:00" level=info msg="log Unmarshal obj: 2 4/usr/local"
```

```
Run: TestReReplication in awesomeGFS
Tests passed: 1 of 1 test - 13 sec 796 ms

Test Results
3 go setup calls
/usr/local/go/bin/go tool test2json -t /private/var/folders/k5/j6y2g8n15tv2b894tttgvz9r0000gn/T/___TestReReplic
2021/07/18 00:19:24 rpc.Register: reply type of method "Mkdir" is not a pointer: "bool"
2021/07/18 00:19:24 rpc.Register: method "ReportChunkServer" has 1 input parameters; needs exactly three
2021/07/18 00:19:24 rpc.Register: method "Shutdown" has 1 input parameters; needs exactly three
2021/07/18 00:19:24 rpc.Register: method "StoreMeta" has 1 input parameters; needs exactly three
time="2021-07-18T00:19:24+08:00" level=info msg="-----new namespace manager-----"
time="2021-07-18T00:19:24+08:00" level=info msg="-----new chunk manager-----"
time="2021-07-18T00:19:24+08:00" level=info msg="-----new chunk server manager-----"
time="2021-07-18T00:19:24+08:00" level=info msg="namespaceManager loading log..."
time="2021-07-18T00:19:24+08:00" level=info msg="log Unmarshal obj: 1 1/test1.txt"
time="2021-07-18T00:19:24+08:00" level=info msg="log Unmarshal obj: 0 1"
time="2021-07-18T00:19:24+08:00" level=info msg="log Unmarshal obj: 1 2/foo"
time="2021-07-18T00:19:24+08:00" level=info msg="log Unmarshal obj: 0 2"
time="2021-07-18T00:19:24+08:00" level=info msg="log Unmarshal obj: 2 3/usr"
time="2021-07-18T00:19:24+08:00" level=info msg="log Unmarshal obj: 0 3"
time="2021-07-18T00:19:24+08:00" level=info msg="log Unmarshal obj: 2 4/usr/local"

Tests passed: 1
```

```
Run: TestMasterLog in awesomeGFS
Tests passed: 1 of 1 test - 568 ms

Test Results
3 go setup calls
/usr/local/go/bin/go tool test2json -t /private/var/folders/k5/j6y2g8n15tv2b894tttgvz9r0000gn/T/___TestMasterLog
2021/07/18 00:19:52 rpc.Register: reply type of method "Mkdir" is not a pointer: "bool"
2021/07/18 00:19:52 rpc.Register: method "ReportChunkServer" has 1 input parameters; needs exactly three
2021/07/18 00:19:52 rpc.Register: method "Shutdown" has 1 input parameters; needs exactly three
2021/07/18 00:19:52 rpc.Register: method "StoreMeta" has 1 input parameters; needs exactly three
time="2021-07-18T00:19:52+08:00" level=info msg="-----new namespace manager-----"
time="2021-07-18T00:19:52+08:00" level=info msg="-----new chunk manager-----"
time="2021-07-18T00:19:52+08:00" level=info msg="-----new chunk server manager-----"
time="2021-07-18T00:19:52+08:00" level=info msg="namespaceManager loading log..."
time="2021-07-18T00:19:52+08:00" level=info msg="log Unmarshal obj: 1 1/test1.txt"
time="2021-07-18T00:19:52+08:00" level=info msg="log Unmarshal obj: 0 1"
time="2021-07-18T00:19:52+08:00" level=info msg="log Unmarshal obj: 1 2/foo"
time="2021-07-18T00:19:52+08:00" level=info msg="log Unmarshal obj: 0 2"
time="2021-07-18T00:19:52+08:00" level=info msg="log Unmarshal obj: 2 3/usr"
time="2021-07-18T00:19:52+08:00" level=info msg="log Unmarshal obj: 0 3"
time="2021-07-18T00:19:52+08:00" level=info msg="log Unmarshal obj: 2 4/usr/local"

Tests passed: 1
```

```
Run: TestMasterCheckPoint in awesomeGFS
Tests passed: 1 of 1 test - 1 sec 698 ms

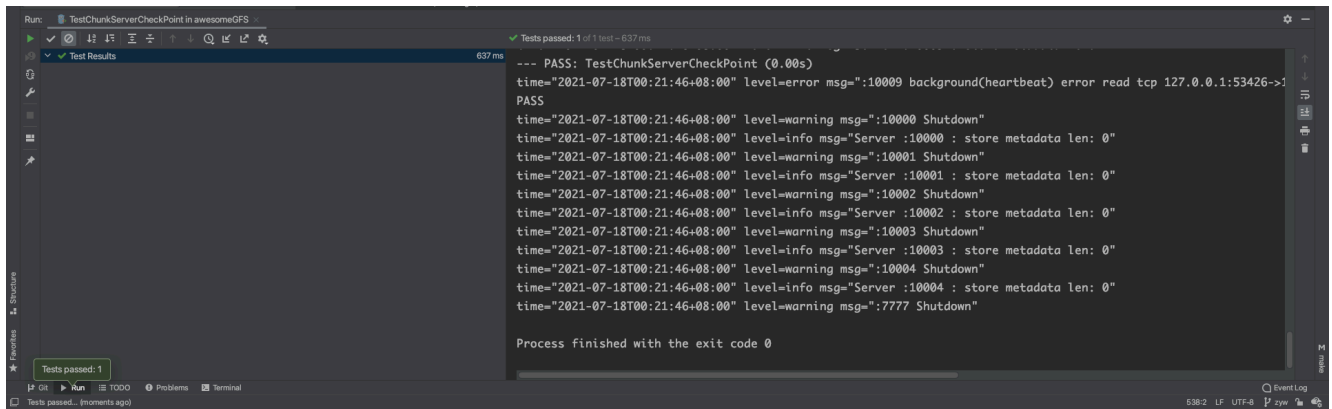
Test Results
3 go setup calls
/usr/local/go/bin/go tool test2json -t /private/var/folders/k5/j6y2g8n15tv2b894tttgvz9r0000gn/T/___TestMasterChe
2021/07/18 00:20:56 rpc.Register: reply type of method "Mkdir" is not a pointer: "bool"
2021/07/18 00:20:56 rpc.Register: method "ReportChunkServer" has 1 input parameters; needs exactly three
2021/07/18 00:20:56 rpc.Register: method "Shutdown" has 1 input parameters; needs exactly three
2021/07/18 00:20:56 rpc.Register: method "StoreMeta" has 1 input parameters; needs exactly three
time="2021-07-18T00:20:56+08:00" level=info msg="-----new namespace manager-----"
time="2021-07-18T00:20:56+08:00" level=info msg="-----new chunk manager-----"
time="2021-07-18T00:20:56+08:00" level=info msg="-----new chunk server manager-----"
time="2021-07-18T00:20:56+08:00" level=info msg="namespaceManager loading log..."
time="2021-07-18T00:20:56+08:00" level=warning msg="open gfs/log/namespace.log: no such file or directory"
time="2021-07-18T00:20:56+08:00" level=info msg="chunkManager loading log..."
time="2021-07-18T00:20:56+08:00" level=warning msg="open gfs/log/chunk.log: no such file or directory"
time="2021-07-18T00:20:56+08:00" level=warning msg="open gfs/log/gfs-master.meta: no such file or directory"
time="2021-07-18T00:20:56+08:00" level=info msg="Master is running now at addr = :7777..."
2021/07/18 00:20:56 rpc.Register: method "Shutdown" has 1 input parameters; needs exactly three
time="2021-07-18T00:20:56+08:00" level=warning msg="Error in load metadata: open /var/folders/k5/j6y2g8n15tv2b85

Tests passed: 1
```

```
Run: TestChunkServerLog in awesomeGFS
Tests passed: 1 of 1 test - 521 ms

Test Results
time="2021-07-18T00:21:31+08:00" level=info msg="Server :10009 : store metadata len: 2"
--- PASS: TestChunkServerLog (0.00s)
PASS
time="2021-07-18T00:21:31+08:00" level=warning msg=":10000 Shutdown"
time="2021-07-18T00:21:31+08:00" level=info msg="Server :10000 : store metadata len: 0"
time="2021-07-18T00:21:31+08:00" level=warning msg=":10001 Shutdown"
time="2021-07-18T00:21:31+08:00" level=info msg="Server :10001 : store metadata len: 0"
time="2021-07-18T00:21:31+08:00" level=warning msg=":10002 Shutdown"
time="2021-07-18T00:21:31+08:00" level=info msg="Server :10002 : store metadata len: 0"
time="2021-07-18T00:21:31+08:00" level=warning msg=":10003 Shutdown"
time="2021-07-18T00:21:31+08:00" level=info msg="Server :10003 : store metadata len: 0"
time="2021-07-18T00:21:31+08:00" level=warning msg=":10004 Shutdown"
time="2021-07-18T00:21:31+08:00" level=info msg="Server :10004 : store metadata len: 0"
time="2021-07-18T00:21:31+08:00" level=warning msg=":7777 Shutdown"

Process finished with the exit code 0
```



最终测试结果为21个测试全部通过。