File consistency in NFS

Ameet Deshpande (CS15B001)

October 19, 2017

1 Checking NFS on DCF Computers

The experiment done on this section was to create two different C files. One of them creates a file called *consistency.txt* and the other reads from the same file. Since we suspect the inode gets refreshed only every $3s$, we expect the second file to give a segmentation fault when it is run as soon as the file is created. The following are the C programs used for the experiments.

```
#include <stdio.h>

int main()
{
    FILE *fp;                                     // Create a file pointer
    fp = fopen(”consistency.txt”,”w”);                // Create a file
    fprintf (fp, ”Operating Systems CS 3500\n”);    // Write to the file
    fclose (fp);                                  // Close the pointer
    return 0;
}
```

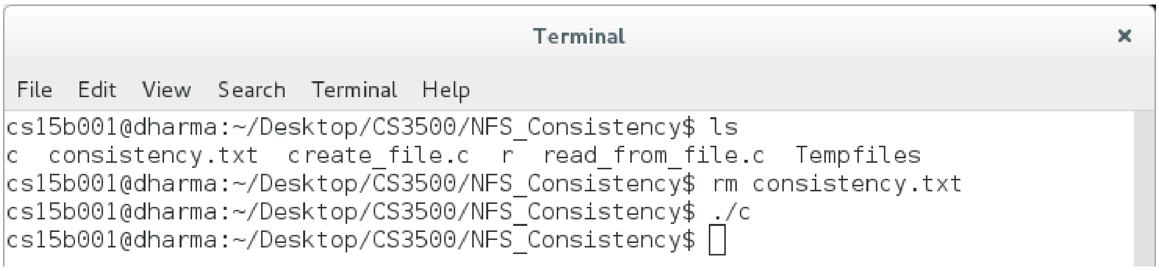
```
#include <stdio.h>

int main()
{
    FILE *fp;                                     // Create a file pointer
    fp = fopen(”consistency.txt”,”r”);                // Create a file

    if(fp == NULL)                                   // Check if the file exists
    {
        printf(”Syncing not done”);
    }
    else
    {
        char ch;
        while(ch != EOF)                          // If file exists, then print the
             contents
        {
            printf (”%c”, ch);
            ch = fgetc(fp);
        }
        fclose(fp);                              // Close the pointer
        return 0;
    }
}
```

2 Results on C files

It is very surprising to see that the second C file prints out the exact output *Operating Systems CS 3500* when both the files are run at almost the same time. How the experiment is performed is, both their respective executables are run at exactly the same time and the output is tested. Even after running this multiple times and with different variation of code, the second file never gives a Segmentation Fault. The expected results were that the file give a segmentation fault. The inode is supposed to get refreshed only once in 3 seconds and the data node once in $30ms$. I suspect the NFS in DCF either has a much higher frequency of reloading the cache, or that since the number of computers are very less in the DCF (active at a time), it can afford to refresh at a much higher rate. There is a $lookupcache = none$ that can be exercised which essentially tells the node to always fetch from the server when needed. These operations can thus be expensive, but for a small network like DCF, it might be reasonable and feasible. It may also be possible that when the request is sent, it checks that the file is not there in local cache. It then checks the server if the file has been created. If so, it fetches it. Howvere commands like *ls* are probably lazy.

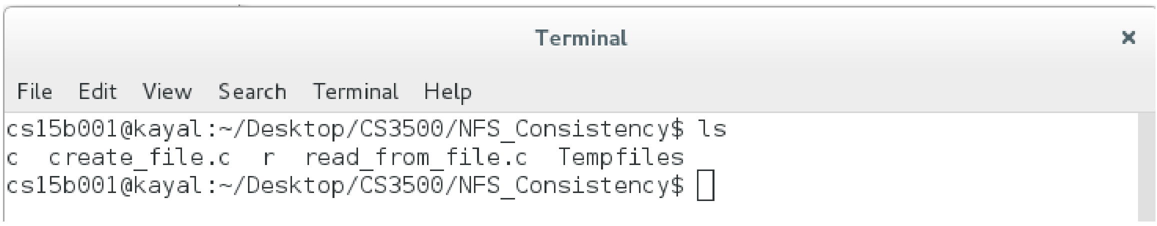
3 Results with ls

The command *ls* however does cause some consistency issues. The following screenshots represent the same.



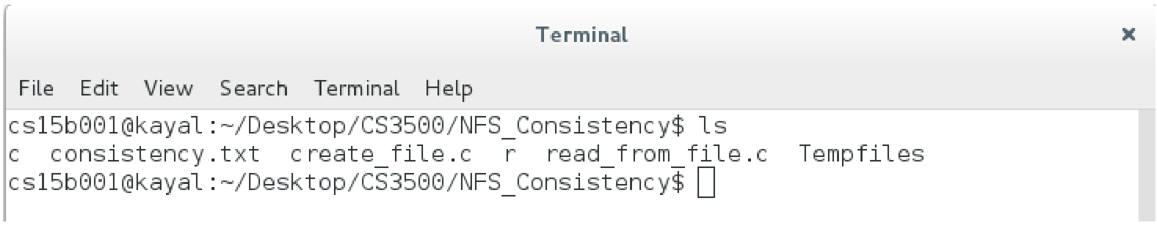
```
cs15b001@dharma:~/Desktop/CS3500/NFS_Consistency$ ls
c  consistency.txt  create_file.c  r  read_from_file.c  Tempfiles
cs15b001@dharma:~/Desktop/CS3500/NFS_Consistency$ rm consistency.txt
cs15b001@dharma:~/Desktop/CS3500/NFS_Consistency$ ./c
cs15b001@dharma:~/Desktop/CS3500/NFS_Consistency$ 
```

Figure 1: Creating the file on one system



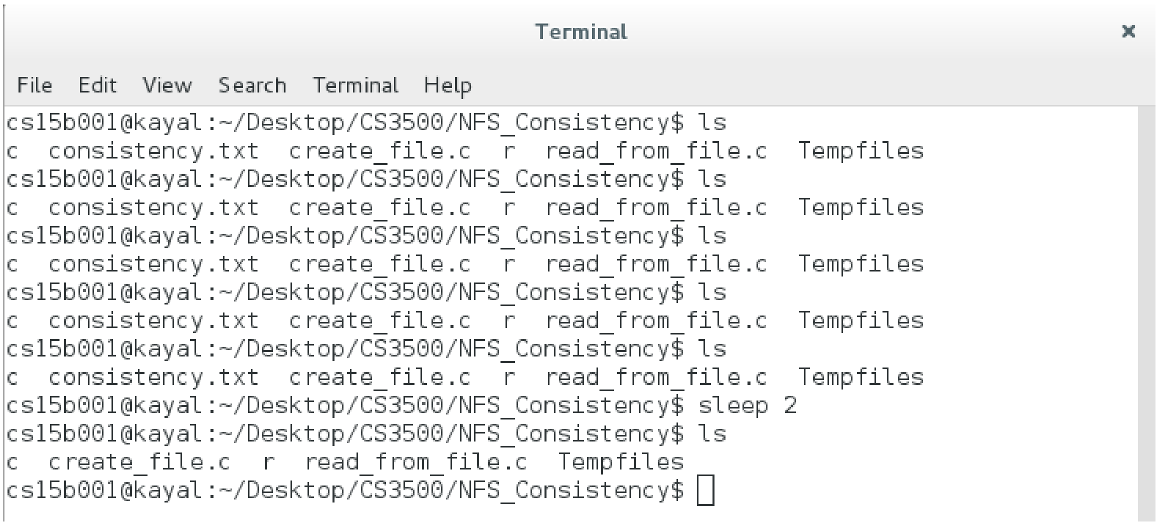
```
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ ls
c  create_file.c  r  read_from_file.c  Tempfiles
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ 
```

Figure 2: Using ls command on the other system



```
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ ls
c  consistency.txt  create_file.c  r  read_from_file.c  Tempfiles
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ 
```

Figure 3: Using ls command after some time



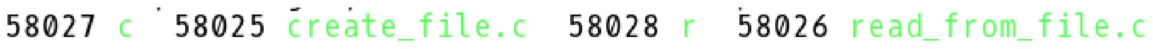
```
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ ls
c  consistency.txt  create_file.c  r  read_from_file.c  Tempfiles
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ ls
c  consistency.txt  create_file.c  r  read_from_file.c  Tempfiles
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ ls
c  consistency.txt  create_file.c  r  read_from_file.c  Tempfiles
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ ls
c  consistency.txt  create_file.c  r  read_from_file.c  Tempfiles
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ ls
c  consistency.txt  create_file.c  r  read_from_file.c  Tempfiles
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ sleep 2
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ ls
c  create_file.c  r  read_from_file.c  Tempfiles
cs15b001@kayal:~/Desktop/CS3500/NFS_Consistency$ 
```

Figure 4: Checking ls after removing the file on the other system

As can be seen, in the second screenshot, the file *consistency.txt* is still not visible. But after a gap of about 3 to 4 seconds, it is visible in Screenshot 3. This clearly means that it is taking a while to sync the system with the server. The fact that the C files work but the *ls* command does not means that *ls* command is probably internally using some other kind of cache which syncs only once every τ seconds. The results are even more dramatic in screenshot 4 where the file *consistency.txt* has been removed on the other system but it is still showing up on this system. However when a *sleep 2* command is executed the file is not shown anymore. This clearly means that the local node is taking a while to sync to the server.

4 Reading the inode of the file directly

Now that we have seen that reading the file from a C executable involves instantaneous sync and using the ls command does provide some latency, we know that the two commands are executed differently. The *ls* command seems to be more lazy in the sense that it takes much longer to sync to the server. There is another command called *ls -i* which basically prints the index nodes of the files along with the filenames. Experimenting with this, I found out that this gives very similar results to the ls commands since it is internally implemented using the ls command. The inode numbers take a while to get refreshed and the results are not instantaneous like in the case of the C files.



```
58027 c   58025 create_file.c   58028 r   58026 read_from_file.c
```

Figure 5: Checking ls -i

5 Using system call to open file

Using the system call does not change anything either. This is expected because the call *fopen* in C uses the system call *open*. Thus, if there is any latency in *open* it will carried in *fopen*. Clearly neither of the calls have any latency in our case.

6 Results

The index node thus does get refreshed only once in a few seconds. We speculate the number of seconds to be more than 1 second and definitely lesser than 5 seconds. The C files are however in my view implemented differently. They are probably checking directly from the server. But the consistency problem with the ls command is very apparent from the screenshots.