Distributed Operating Systems

Exercise 2: Distributed File Systems

In the tutorial, all solutions will be presented by students. Please be prepared for all questions as the exercise will focus on discussion. Some questions require knowledge that was not presented in the lecture. Reading material is listed below, in particular, you should read the paper describing the Google file system (GFS).

The exercise aims at understanding how scalability can be achieved in distributed file systems and what challenges distributed storage causes for fault tolerance. The first part covers the SUN Network File System (NFS), which is a representative of "classic" distributed file systems. Its basic design was covered in the basic OS lecture, but it is also described in the books referenced at the bottom of this page. In the second part the much more scalable Google File System is discussed.

The SUN Network File System

1) You use a shell script (download link: test.sh) that creates and deletes files and directories in an NFS-mounted file system. After each operation, the script checks the status of the file or directory on another computer that has NFS-mounted the same filesystem. You get the following screen output:

```
user@ganymed:~$ sh test.sh serv9
localhost: creating /usr/users/sya/cw183155/tmp-filename and
checking on serv9 for existence
serv9: file exists
localhost: writing data to /usr/users/sya/cw183155/tmp-filename and
checking file size on serv9
serv9: file still contains zero bytes
serv9: file still contains zero bytes
serv9: file still contains zero bytes
serv9: file contains some data
localhost: removing /usr/users/sya/cw183155/tmp-filename again and
checking on serv9 for existence
serv9: file still exists
serv9: file still exists
serv9: file still exists
serv9: file is gone
localhost: creating directory /usr/users/sya/cw183155/tmp-filename
and checking on serv9 for existence
serv9: directory exists
localhost: removing directory /usr/users/sya/cw183155/tmp-filename
and checking on serv9 for existence
serv9: directory still exists
```

```
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory still exists
serv9: directory is gone
user@ganymed:~$
```

Explain these results using your knowledge of the NFS architecture.

The Google File System

The Google File System (GFS) has been specifically designed with scalability in mind. The design and motivations behind it are explained comprehensively in the 2003 research paper "The Google File System". It discusses many interesting aspects that could not be covered in the lecture. Please read it, so that you can better answer the questions below and be prepared for the discussion during the exercise.

2) **GFS design choices:** What are the major design choices made by the Google engineers? How are they motivated and what are the consequences? Compare your findings to what you know about NFS.

3) **Scalability techniques in GFS:**

   a) Describe how GFS can scale to many hundreds (or more) machines despite having a single master.

   b) Describe the caching mechanisms used in GFS.

   c) What is the benefit of "record append"?

4) **Fault tolerance in GFS:**

   a) How does GFS protect itself against a failing master server?

   b) What happens, if one or more chunkservers fail? Describe the recovery procedure.

   c) Can clients read stale data?

   d) What happens, if a client fails?

e) How does GFS achieve fault tolerance against disk failures? Consider chunkservers and the master.

f) Explain the consistency model of GFS. What does it mean for applications?

5) **Workloads, applications, fun stuff:** GFS may not be suitable for certain applications. What kind of workloads would you consider problematic and why? How could a better file system for your favorite application look like? Be creative!

Material

BOOKS

- A. Tanenbaum, "Distributed Operating Systems"

- G. Coulouris et. al., "Distributed Systems, Concepts and Design"

- Research paper: Ghemawat et. al., "The Google File System"

SOFTWARE

- Shell Script test.sh for exercise 1)