Google FS (GFS)

Goal: support Google workloads

=> contrast to "general purpose" file system differences?

no latency concerns (scalable bandwidth)

no POSIX legacy apps (they're also writing apps)

only certain W/L charac.

  e.g. large files, streaming

Assumptions

failure: common

"few" huge files (not many small)

large streaming reads

seq writes: esp. append

  e.g., no need for bandwidth (not latency)

=> design decisions

for client caches
Interface: (to clients)
looks like normal "FS"
open/read/etc.
append: commonly used

General Arch
radical: single master
(single point of failure)
follow-on: Colossus
adopted "multi master"

network

clients

master
chunk servers

M

CS

C1

Cm

most of what you do
heartbeat
roots: CMU NASD '98

File: ≥ 1 chunks
  each: has 1D (64-bit) ID
  [64MB]
Chunk servers: 3 replicas
  => read/write of chunks (offset, range)
  => on top: Linux FS

Advantages:
  - easier (leverage)
  - user-level service
  - abstraction of local file
  - chunk \rightarrow\ file
    (64MB) supports 'sparse' ness

Client: normal "fs" =) syscalls
  GFS =) client library
client lib => RDCs => master => CS's

no caches

master:
all metadata => all decisions

Chunk Size: 64 MB

=> [no internal frag] => local FS

could have smaller block instead
but... problems:
- more metadata (@ master)
- more interaction w/ master
- more metadata (@ clients)
Problems w/ 64MB block:

=") hot spot: "small" file (< 64MB) is popular

"solution: replicate more:
N ∼ 20

Start @ 1:43/1:44

Admin:

=") Projects

=") one more meeting (Mon eve.)

=") Final pres.:
~20 minute
12/15 Fri or
12/20 Wed

=") short writeup (5 pages)

=") Midterm
Exam
take-home
email:

("12 hr") [Sun 17th]
[Mon 18th]
(not collaborative)
Exam Goals: sleep

Non paper writing strategy

---

Master: Metadata = master memory
   benefits: centralized, fast

Types of metadata:
- file/chunk namespaces
- map: file -> chunk ID
- map: (ID -> replicas)
  want: small
  stored at CS's
  => checkpoint periodically
Me \leftarrow \text{heartbeat} \rightarrow \text{CS} \ldots \text{CS}

\underline{Consistency} \quad \underline{mutations}

\underline{Namespace:} \rightarrow \text{master}
\underline{mutate:} \text{easy} \ (\text{locks, do it})
\rightarrow \text{atomic}

\underline{File Data:}
\underline{consistent:} \ all \ clients \ see \ same \ data \ (regardless \ of \ replica \ read)
\underline{defined:} \ consistent \ and \ see \ all \ results \ of \ entire \ client \ mutations
**Example**: "Defined"

- $P_1$: `write`
- $P_2$: either `write` or `write` in entirety

**Diagram**:

```
File
```

<table>
<thead>
<tr>
<th></th>
<th>writes</th>
<th>appends</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial</td>
<td>defined (cons.)</td>
<td>defined (but interspersed or</td>
</tr>
<tr>
<td>success</td>
<td></td>
<td>in consistent)</td>
</tr>
<tr>
<td>conc.</td>
<td></td>
<td>cons. (undefined)</td>
</tr>
<tr>
<td>success</td>
<td></td>
<td></td>
</tr>
<tr>
<td>failure</td>
<td></td>
<td>in consistent</td>
</tr>
</tbody>
</table>
Reads :

C \xrightarrow{m} m

\(\mathbb{C}^{}\)

Mutations : writes / Appends

writes: use primary/backup

(serialize)

lease: master \rightarrow CS

Flow : 3 phase

1) C \xrightarrow{m} m \quad \text{metadata}

2) C \xrightarrow{CS_1, \ldots, CS_3} \quad \text{data} \quad \text{num + ack}
pipeline not \( 1 \rightarrow \text{many} \)
\[
\begin{align*}
C & \rightarrow CS_1 \\
& \rightarrow CS_2 \\
& \rightarrow CS_3
\end{align*}
\]

3) \( C \rightarrow CS_{\text{primary}} \rightarrow CS_{\text{sec}} \) (in order)

\( \text{ack} \rightarrow CS_{\text{sec}} \)

notes:
- write can fail
- \( \Rightarrow \) retry
- fail \( \Rightarrow \) inconsistent
- big requests \( \Rightarrow \) undefined

64MB
Appends:
except:
  primary: has to decide
       \[\text{which offset to write}\]
  secondaries

\[\text{inconsistent } R_1 \quad R_2 \quad R_3\]
\[\Rightarrow \text{defined}\]
\[\Rightarrow \text{duplicates}\]
\[\Rightarrow \text{left to } APP\]