Operation	Read		Write		Record Append	
Cluster	X	Y	X	Y	X	Y
0K	0.4	2.6	0	0	0	0
1B..1K	0.1	4.1	6.6	4.9	0.2	9.2
1K..8K	65.2	38.5	0.4	1.0	18.9	15.2
8K..64K	29.9	45.1	17.8	43.0	78.0	2.8
64K..128K	0.1	0.7	2.3	1.9	< .1	4.3
128K..256K	0.2	0.3	31.6	0.4	< .1	10.6
256K..512K	0.1	0.1	4.2	7.7	< .1	31.2
512K..1M	3.9	6.9	35.5	28.7	2.2	25.5
1M..inf	0.1	1.8	1.5	12.3	0.7	2.2

Table 4: Operations Breakdown by Size (%). For reads, the size is the amount of data actually read and transferred, rather than the amount requested.

and file systems, but the effect is likely more pronounced in our case.

6.3.2 Chunkserver Workload

Table 4 shows the distribution of operations by size. Read sizes exhibit a bimodal distribution. The small reads (under 64 KB) come from seek-intensive clients that look up small pieces of data within huge files. The large reads (over 512 KB) come from long sequential reads through entire files.

A significant number of reads return no data at all in cluster Y. Our applications, especially those in the production systems, often use files as producer-consumer queues. Producers append concurrently to a file while a consumer reads the end of file. Occasionally, no data is returned when the consumer outpaces the producers. Cluster X shows this less often because it is usually used for short-lived data analysis tasks rather than long-lived distributed applications.

Write sizes also exhibit a bimodal distribution. The large writes (over 256 KB) typically result from significant buffering within the writers. Writers that buffer less data, checkpoint or synchronize more often, or simply generate less data account for the smaller writes (under 64 KB).

As for record appends, cluster Y sees a much higher percentage of large record appends than cluster X does because our production systems, which use cluster Y, are more aggressively tuned for GFS.

Table 5 shows the total amount of data transferred in operations of various sizes. For all kinds of operations, the larger operations (over 256 KB) generally account for most of the bytes transferred. Small reads (under 64 KB) do transfer a small but significant portion of the read data because of the random seek workload.

6.3.3 Appends versus Writes

Record appends are heavily used especially in our production systems. For cluster X, the ratio of writes to record appends is 108:1 by bytes transferred and 8:1 by operation counts. For cluster Y, used by the production systems, the ratios are 3.7:1 and 2.5:1 respectively. Moreover, these ratios suggest that for both clusters record appends tend to be larger than writes. For cluster X, however, the overall usage of record append during the measured period is fairly low and so the results are likely skewed by one or two applications with particular buffer size choices.

As expected, our data mutation workload is dominated by appending rather than overwriting. We measured the amount of data overwritten on primary replicas. This approximates the case where a client deliberately overwrites previous written data rather than appends new data. For cluster X, overwriting accounts for under 0.0001% of bytes mutated and under 0.0003% of mutation operations. For cluster Y, the ratios are both 0.05%. Although this is minute, it is still higher than we expected. It turns out that most of these overwrites came from client retries due to errors or timeouts. They are not part of the workload *per se* but a consequence of the retry mechanism.

Operation	Read		Write		Record Append	
Cluster	X	Y	X	Y	X	Y
1B..1K	< .1	< .1	< .1	< .1	< .1	< .1
1K..8K	13.8	3.9	< .1	< .1	< .1	0.1
8K..64K	11.4	9.3	2.4	5.9	2.3	0.3
64K..128K	0.3	0.7	0.3	0.3	22.7	1.2
128K..256K	0.8	0.6	16.5	0.2	< .1	5.8
256K..512K	1.4	0.3	3.4	7.7	< .1	38.4
512K..1M	65.9	55.1	74.1	58.0	.1	46.8
1M..inf	6.4	30.1	3.3	28.0	53.9	7.4

Table 5: Bytes Transferred Breakdown by Operation Size (%). For reads, the size is the amount of data actually read and transferred, rather than the amount requested. The two may differ if the read attempts to read beyond end of file, which by design is not uncommon in our workloads.

Cluster	X	Y
Open	26.1	16.3
Delete	0.7	1.5
FindLocation	64.3	65.8
FindLeaseHolder	7.8	13.4
FindMatchingFiles	0.6	2.2
All other combined	0.5	0.8

Table 6: Master Requests Breakdown by Type (%)

6.3.4 Master Workload

Table 6 shows the breakdown by type of requests to the master. Most requests ask for chunk locations (*FindLocation*) for reads and lease holder information (*FindLease-Locker*) for data mutations.

Clusters X and Y see significantly different numbers of *Delete* requests because cluster Y stores production data sets that are regularly regenerated and replaced with newer versions. Some of this difference is further hidden in the difference in *Open* requests because an old version of a file may be implicitly deleted by being opened for write from scratch (mode “w” in Unix open terminology).

FindMatchingFiles is a pattern matching request that supports “ls” and similar file system operations. Unlike other requests for the master, it may process a large part of the namespace and so may be expensive. Cluster Y sees it much more often because automated data processing tasks tend to examine parts of the file system to understand global application state. In contrast, cluster X’s applications are under more explicit user control and usually know the names of all needed files in advance.

7. EXPERIENCES

In the process of building and deploying GFS, we have experienced a variety of issues, some operational and some technical.