1 Summary

The Google File System (GFS) was Google's response, at the time, to its growing data processing needs and its, then, radical choices in the design space like accepting component failures as the norm rather than exception and large file sizes. This GFS paper is probably one of the first papers, first at least for Google, that applies typical notions of ditributed computing at such massive scale and that too, in industry. The success of relaxing consistency to improve performance and simplify design further enforces the notion of being well-advised to do so compared to relaxing availability or partition. In terms of technical contributions, this paper presents file system interface extensions, discusses various aspects of their system design, and reports measurements based on benchmarks and real-world use.

2 Strengths of the paper

At the time this paper was written, the authors' reexamination of traditional choices and exploration of radically different points in the design space is what led to the success of GFS and paved the way for modern equivalents. These new points included:

1. Thinking about component failures as the norm rather than the exception
2. Accepting that files are huge by traditional standard (e.g., mutli-GB)
3. Files are mostly changed by appending new data rather than overwriting existing data
4. It is beneficial to co-design the applications and file system API as it increases flexibility

There are several other great ideas in this paper that form the basis of many modern-day distributed file-systems such as, but not limited to:

- decoupling metadata and data operations
- use of checkpoints for consistency and efficient state recovery
- file chunking for parallel throughput

3 Weakness of the paper

I genuinely am at a loss for words here. This is such a good technical paper. Maybe, just maybe, the paper could have demonstrated or mentioned something about segregating masters based on some constraints to incorporate sharding at an even higher-level?

4 Future work opportunities

The authors note that GFS aims to suit specific sort of workloads. These include heavy reads and writes, concurrent heavy appends, large data processing (requiring high, sustained bandwidth), and working on large files. How about a semi-opposite workflow, especially one with more random rather than serial access?