Paper title: Speculative Execution in a Distributed File System
Paper authors: Edmund B. Nightingale, Peter M. Chen, Jason Flinn
Your name: Parth Kansara (115135130)

What problem does the paper address? How does it relate to and improve upon previous work in its domain?
For distributed file systems, the performance is usually lower than that of local file systems due to the high number of I/O operations required for cache coherence. This is aggravated by increased latency in the network due to several factors. Distributed systems like NFS trade in consistency for better performance, but even the weaker consistency semantics require a higher overhead. Thus, lower performance and weaker consistency are common problems in distributed file systems, and are addressed in this paper.

Chang & Gibson and **Fraser & Chang** implement speculative execution for a local file system, to utilize prefetching for improving read performance. Speculator, introduced by the paper, improves read performance by lowering the cost of cache coherence. Further, the previous systems require the computations of speculative threads to be redone by the non-speculative threads. Along with that, the speculative threads are restricted from communicating with other processes.

Time Warp looks at speculative execution for distributed simulations, but has certain restrictions like only allowing deterministic processes, not allowing the usage of heap storage, and communication via asynchronous messages. Unlike Time Warp, Speculator is able to utilize the client-server nature of the distributed file systems to simplify the speculation process. Several works like **Steffan et al.** looked at processor-based speculation, and its utility to obtain greater parallelism from applications. However, speculation at processor-level does not provide the same kind of higher-level abstractions as the OS-level speculation. Earlier distributed file systems like **Sprite** and **Spritely NFS** offered single-copy file semantics, which had a performance overhead. **Liskov** and **Rodrigues** showed that by allowing clients to read slightly stale data, read-only transactions can be fast. Speculator builds on top of these and provides strong safety and cache coherence guarantees while having very low impact on performance.

What are the key contributions of the paper?
The paper addresses the above issues faced by distributed systems and presents the Speculator - Linux kernel support for speculative execution. This idea of speculative execution in distributed systems roots from three observations - filesystem clients can accurately predict the result of several operations, a network RTT is effectively longer than the time taken for a lightweight checkpoint which provides time to finish tasks while waiting for a request and lastly, the resources required to execute processes speculatively can be spared by modern systems. The designed Speculator restricts processes that are executing speculatively to externalize output before the speculations are not proved true. It also blocks any irreversible operations by speculative processes till the speculations are not proved true. The Speculator does a copy-on-write fork of the current process for checkpointing and discards the checkpoint if all the speculations are true, else it restores the state of the process at the checkpoint and terminates it to prevent any externally visible outputs. It also provides two structures, *speculation structure*, which notes all the kernel objects dependent on the new speculation and the *undo log*, which tracks all the modifying operations on the object that can be reversed in case of a rollback.

A correct speculative execution is defined by two conditions - speculative state should be hidden externally and non-speculative processes should not be able to view speculative state. Speculator extends to multi-process speculative execution as well, by allowing speculative processes to engage in IPC. Speculator also supports speculative states in local memory and disk file systems by modifying the RAMFS and the ext3 file systems respectively. It can also track speculative dependencies that are passed through pipes, fifos, Unix sockets, signals, forks and exits. In a distributed file system, the Speculator was implemented to perform a speculative mutation on the file, only if the underlying speculation hypothesis is true. It also provides for group commits and allows a single thread to have several synchronous mutating operations, which improves throughput.

Briefly describe how the paper's experimental methodology supports the paper's conclusions.
The Speculator provides a significant boost in the performance of distributed file systems. The PostMark and Andrew-style benchmarks demonstrate a 2x improvement in performance of NFS over LAN. The paper also modifies the Blue File System to integrate Speculator which provides strong guarantees of safety and consistency while performing 66% faster than the original NFS. These observations demonstrate the capabilities of the Speculator and serve as a proof of its claim of enhancing performance while maintaining strong guarantees for safety and consistency.

Write down one question you may want to ask the authors.
Is there any possibility of integrating low-latency ML solutions for the speculations done in the Speculator?