## Paper summary

Summary: In a distributed environment, consistency can cause a significant overhead for performance. A strong consistency requires a lot of synchronous calls. Even weaker consistency like close-to-open consistency requires multiple synchronous calls. The Speculator assumes that there will be a performance gain if the hypothesis holds, or it will roll back with a lower penalty overhead. Hence, it performs sync operation in an async manner. The process should be highly predictable for a thriving distributed environment, checkpointing should be quicker than remote operation, and resources should be available in spare for the Speculator to speculate. Speculation is invisible to the user, and the process and Speculator have no idea why the client speculates. It can allow two mutating processes to share a joint checkpoint, but performance results don't support the concept. In multiprocess belief, the process cooperates. Speculator tracks causal dependencies between kernel objects to share speculative state among multiple processes. Speculation makes group commits hence reducing various round trips. Evaluation of Andrew and PostMark benchmarks for NFS, and single-copy file semantics and synchronous I/O (in BlueFS) is provided. From the benchmarks, we can conclude that Speculator improves existing distributed file systems by enabling them to be safe, fast, and consistent.

## Strengths

Speculator exists on top of the Linux kernel hence POSIX compliant. Simple abstraction for the client. Invisible to the user and the process it is not associated with. Minimalistic changes (only 7500 lines of code). The paper takes inspiration from transactions and rollbacks from databases, branch prediction, and thread-level multicore speculation. (use a good idea again)

## Weaknesses

Vulnerable to security attacks such as Spectre and Meltdown. (Source: https://en.wikipedia.org/wiki/Speculative_execution#Security_vulnerabilities). Logging can be difficult in delayed and group commits. Some more benchmarks exploring diverse scenarios would be more desirable.

## Comments for author

How scalable is this system?