Cover Letter: Cudele (Paper 306)

We thank the reviewers for their hard work and genuinely helpful suggestions. In addition to this cover letter, we have posted a PDF online explicitly showing additions in **blue** and deletions in **red**. Three issues were common across reviewer feedback; Reviewer 1 and 3's comments are addressed in issues II and III, Reviewer 2's comments are addressed in all issues, and Reviewer 4's comments are addressed in issues I and II.

ISSUE I: IRRELEVANT AND/OR SYNTHETIC USE CASES (REVIEWERS 2 AND 4)

We re-wrote Section §V-B to highlight examples from HPC and cloud workloads that would benefit from using Cudele. For HPC we focus on user home directories used for experiments and for the cloud we focus on the Hadoop and Spark runtimes (as suggested by Reviewer 2). We also added **Cudele Setup** headers in Section §V-B to show which subtree semantics accommodate each workload and to demonstrate how Cudele might be able to help in these scenarios. By showing these setups, we hope to demonstrate that different applications can use a file system in parallel with different consistency/durability semantics. Although we do not actually run the workloads on our prototype, hopefully our changes show how the use cases, and the synthetic benchmarks we used to represent them, are indeed relevant to parallel and distributed computing.

To align the paper to the themes of the conference, we changed Section §V-A and Figures 6a, 6b, and 6c to emphasize that decoupled namespaces facilitate client-driven parallelism, where clients detach subtrees form the global namespace and do operations in-parallel to their local disk. We also hope that changes to the use cases above motivate the need for robust distributed storage systems that can handle today's applications. So while the paper is storage-centric, we try to show that the workloads it supports are highly distributed and need a flexible solution like Cudele.

ISSUE II: STRUCTURE AND LAYOUT OF EVALUATION (ALL REVIEWERS)

To clarify the evaluation and eliminate some of the confusion due to our cross-referencing, we:

- connected performance gains from the abstract and Section §I to Section §V to clarify speedup/slowdown comparisons and to show how results are derived; we also re-labeled all graphs with the same metric (throughput slowdown/speedup) and aligned Section §V with the graphs in Section §II.
- made experiments self-contained in Sections §II and §V by removing cross-references and adding baseline values so readers can calculate raw numbers from our speedup/slowdown graphs. We re-ran all experiments to verify the raw numbers and tweaked the visual representations of the graphs for clarity (although the actual results are the same).
- removed "major takeaways" from Section §V and focused on insights into results by analyzing raw numbers in comparison to hardware speeds.
- removed future work from Section §I to make the contributions explicit. We do not attempt to validate or prove any of the statements about the benefits of changing consistency and durability properties of a subtree dynamically, such as saving resource provisioning costs.
- clarified confusing terminology in Section §I: administrator refers to a person that configures subtrees and client refers to a storage client or application that interacts with the metadata server. We avoid using the term "user" except when referring to end-users that interact with home directories in a file system or the Hadoop/Spark runtime itself.

ISSUE III: NO EXPERIMENTAL SETUP (REVIEWERS 1, 2, AND 3)

We apologize for the omission of experimental setup and source code details. For the experimental setup, we describe the cluster setup (hardware, software, etc.) in Section §V. We have also made the infrastructure code available and added links after each figure to show exactly how experiments are run. This infrastructure code contains scripts to deploy the system, run experiments, and gather results. This process follows the Popper Convention, which aims to make research reproducible. We also made the source code available online and a link is provided in Section §V. Finally, we add details about which tools and code bases we modified in Section §IV to address Reviewer 1's questions about the framework implementation.

ADDITIONAL COSMETIC FIXES

For Reviewer 1, we fixed user/client terminology in Section §III, clarified that the cost of dynamically changing semantics is future work in Sections §I and §III, and added source code pointers in Section §V. For Reviewer 2, we removed the major takeaway headings and cross-references in Section §V. For Reviewer 3, we quantified speedups with figure annotations in Section §V. For Reviewers 1 and 3, we added servers, network, and storage setups in Section §V. For Reviewer 4, we added descriptions to each use case in Section §V describing how Cudele would work with Spark.