Cover Letter

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 **red** and deletions in **blue**. Three issues were common across reviewer feedback; Reviewer 1 and 3's comments are addressed in issues I and II, Reviewer 2's comments are addressed in issues I and III, and Reviewer 4's comments are addressed in issues II and III.

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

We apologize for the omission of experimental setup and source code details. To provide a more comprehensive view of our experiments, we added text to the paper and made paper artifacts available online. 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[1] [1], which aims to make research reproducible. We also made the source code available online and a link is provided in a footnote in Section §V. Finally, we add details about which tools and classes we modified in Section §IV to address Reviewer 1's questions about the framework implementation.

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

All reviewers note that the results and contributions we cite in the introduction are not validated, explained, or even mentioned in the evaluation. One confusing component of this issue is that we mixed future work with the contributions of the paper; to make the contributions more explicit we remove future work from the introduction and add it to Section §VI; 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. To further clarify the evaluation, we:

- explicitly state the Cudele setup and include baseline values in the text so readers can quickly calculate raw numbers from our speedup/slowdown graphs. We also make sure to connect all speedups from the introduction to the evaluation. We hope this clarifies confusion about what we are comparing to when we reference speedups and slowdowns.
- re-organized the sections and removed cross-references. Experiments are now self-contained so the reader can see the effects of different API configurations and we do a better job of explaining how results are derived.
- removed the "major takeaways". We deleted the headings but moved the text to the beginning of each section to motivate why we are doing each experiment. We also add insights into the results by analyzing the raw numbers we observe in comparison to hardware capabilities.
- re-labeled all graphs with the same metric (throughput slowdown/speedup) and aligned Section §V with the graphs in Section §II. We hope this eliminates more of the confusion triggered by the cross-referencing we did, referenced by Reviewers 2 and 3.

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

In the evaluation we add examples from HPC and cloud workloads that demonstrate how our benchmarks are representative of problems in real-world systems; we actually take Reviewer 2's advice of using Spark as an example and show where and how these workloads benefit from Cudele in each use case in Section §V-B. We are also clearer about how Cudele's role in addressing the problem by showing how we configured subtrees with different semantics to accommodate these workloads in the global namespace. These are marked with **Cudele setup** headers in Section §V-B. 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 are indeed relevant to parallel and distributed computing.

This emphasis on relevant use cases in HPC and the cloud should also show the need for parallel and distributed storage systems that keep up with the computation, whether the applications are MPI-based, MapReduce-based, etc. 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. Regardless, we do make an effort to better highlight the parallel and distributed computing themes in Cudele in Section §V-A and the axis labels in Figures 6a, 6b, and 6c. Namely we emphasize that decoupled namespaces allow concurrent clients in the global namespace to detach subtrees and do operations in-parallel on their local disk.

[1] http://falsifiable.us/

COSMETIC FIXES

- Reviewer 1: fixed user vs. client terminology in Section §III
- Reviewer 1: clarified that the cost of dynamically changing semantics is future work Sections §III and §V.E
- Reviewer 1: added source code pointer in Section §IV
- Reviewer 2: remove major takeways in Section §V
- Reviewer 2: remove major cross-references in Section §V
- Reviewer 3: quantified speedups with figure annotations in Section §V
- Reviewers 1 and 3: add servers, network, and storage setups in Section §V
- Reviewer 4: add new section (§V.F) describing how Cudele would work with a system like Spark

REFERENCES

[1] I. Jimenez, M. Sevilla, N. Watkins, C. Maltzahn, J. Lofstead, K. Mohror, R. Arpaci-Dusseau, and A. Arpaci-Dusseau, "Popper: Making Reproducible Systems Performance Evaluation Practical," UC Santa Cruz, Technical Report UCSC-SOE-16-10, May 2016.