From: Arif Merchant, aamerchant@google.com
To: Agnieszka Augustyniak, University of Warsaw, Scientific Councils Office

Subject: Report on dissertation draft: Adapting Distributed Storage with Deduplication to Cloud Use Cases

To whom it may concern:
I have reviewed the draft of the dissertation "Adapting Distributed Storage with Deduplication to Cloud Use Cases". My overall evaluation is that **I deem the thesis as sufficient to grant a PhD.** I add some detailed comments below in the hope that the student will find them useful in further improving the draft, but for the most part, the dissertation is acceptable as is.

Summary: The dissertation combines together three substantial pieces of research, each of which has been published in a prestigious venue, plus some material to bind the work together. I read through the dissertation in reasonable detail, and have no concerns about rating this as a "Pass". There are, of course, things that can be improved, whether in the description of the work or in the evaluations. I have noted my comments below, in the hope that the student will find helpful nudges for improvement, and I hope the student will implement those that are possible, but for the most part, the dissertation is acceptable as is.

Chapters 1-3: I found Chapters 1-3 to be generally well-written and quite interesting. I do not have many comments on the material there. Mainly, I wonder how some of the material is relevant to the subject of this work: for example, Section 3.4, on the Object Store Interface. It would be good to connect up the material better with the rest of the dissertation, but I have no significant concerns there, so I will focus mainly on the remaining chapters.

Chapter 4 Summary: This chapter is dedicated to the description of ObjDedup, an Object-based storage system layer above a block store (HYDRAstor) that internally implements block-based deduplication. The focus is on providing deduplicated storage with an Object API, such as would be needed for backup and similar applications. The dissertation provides background, including related work, a description of the principles behind the design of ObjDedup, a brief description of the implementation, and an evaluation using microbenchmarks on an actual implementation.

Overall, the writing is clear and well done. The related work cited is fairly extensive, and I did not notice any obvious gaps, There are a few areas where more clarification might be helpful, however:

- The question that begs to be answered is: what does the ObjDedup design and implementation have to do with deduplication? The actual deduplication is implemented in a lower layer, HYDRAstor, whose design is not a part of this dissertation (and actually appears to be a commercial product). Since this layer presents a simple block interface (please clarify if this is incorrect!), the Object storage layer could run on top of any block store. So how exactly is ObjDedup designed to take advantage of the deduplication built into HYDRAstor? It appears that ObjDedup is write-optimized, particularly with respect to

Object metadata, but that has little to do with deduplication, and more to do with the kinds of workloads it is intended to support.

- The evaluation is based mostly on microbenchmarks, intended to show that ObjDedup scales well, whether with object size, the number of objects, the ingress rate, etc. It would be preferable if there were also some evaluation with macro-benchmarks or workload traces, particularly if traces from supported application workloads are available.
- It would also be preferable if, instead of just an internal evaluation using just ObjDedup+HYDRAstor, we had a comparison with alternative object stores. I understand from the background description that there may not be alternatives where an object store is implemented atop a deduplicated block storage, but presumably there are some implemented on non-deduplicated block storage? Or perhaps dedicated object stores? Ceph comes to mind, but the author may be able to come up with alternatives that are easier to compare against or that support similar workloads.
- Some additional minor comments:
    - Section 4.2.2 talks of real-world deployments but then does not provide any details at all regarding these deployments. I understand there may be confidentiality concerns for exposing the full details, but it should presumably be possible to give broad descriptions and statistical characterizations.
    - In the evaluation, is it possible to test how ObjDedup affects the deduplication performance? I would expect it to have minimal impact, but can you verify? Or even show that the deduplication ratio improves compared to alternatives?

Chapter 5: InftyDedup Summary: This chapter presents InftyDedup, a system for tiering local and cloud storage, with separate deduplication in each tier. The cloud tier may actually have multiple parts, with different cost, bandwidth, and latency properties. InftyDedup tries to move data around to reduce cost while still meeting the storage requirements of the data.

The description of this system is not entirely crisp, Here are some of the issues that are unclear to me:

- Is this application only intended for backup data, or is this only a particular focus? If it is intended for backup data only, some of the questions below may not apply.
- What is the granularity of upload/data movement? Block? File? Groups of files? I would guess block, based on the description, though it isn't entirely clear. In any case, how is this decision made? It does matter, for multiple reasons:
    - If the granularity is fine (block) then there is likely metadata blow-up. This is possibly not apparent in a small-scale deployment, but once the number of objects grows to 10^11 or more, managing the metadata can be a major issue. At least, there needs to be an explicit decision that the system is only up to a certain size, and an exploration of the metadata complexity.
    - If the granularity is large (say, a group of files or a volume), then metadata scale issues are abated, but deduplication efficiency might take a hit.
- What is the frequency and schedule of data upload into the cloud? This has to depend on resource availability, including upload bandwidth, to avoid interfering with foreground workload requirements. It also has to depend on the workload characteristics of the data

to be uploaded: for example, you may want to keep hot data local for longer; and since GC can be expensive in the cloud, perhaps delay the upload of data likely to be deleted soon?

- How does batch deduplication scale? As the data archived in the cloud grows larger and larger, it would seem that matching fingerprints might become expensive.
- It is unclear how the hot/cold storage method works with deduplication. While it is conceivable that, given the restore frequencies, one can optimize whether to place the data in hot or cold storage in the cloud, how does this work when a block could, through deduplication, be a part of multiple different files, all with different restore frequencies? And it is unlikely that restore frequencies can be predicted very accurately in general, so what is the impact of inaccurate estimates in this regard?
- In the evaluation, do the incremental backups also have considerable duplication? If so, how is that justified? One would expect the incremental changes to be new and different data. But perhaps there are measurements in the field that show there still is duplication.
- In the evaluation, it is not clear how well InftyDedup scales. If you have any additional data to demonstrate scaling better, it would be interesting to see.

Chapter 6: Derrick Summary: This chapter deals with Derrick, a system comprising three sub-components (CentralBal, TrBal, and DistrBal) to control the placement and rebalancing of the data in the system. There is some background, a high-level description of the components, and an evaluation where the performance is compared against similarly targeted algorithms in Ceph and Swift. One of the components is directed at general placement and rebalancing, another to control the actual movement process, and the third deals with emergency movements engendered by hardware failures.

- Overall, I found this chapter a bit opaque and hard to read, which was a little surprising considering it has already gone through review with ACM ToS. Some of the central concepts were only loosely defined - for example, group, or component of a group. This made it difficult to understand the rest of the chapter. Note that "group" is not an entirely standard term in this context. From the description, it sounded like an aggregation construct, like a logical volume, but the examples were much smaller than a typical logical volume. Some related systems like Pacemaker/Tiger use "group" to mean a "redundancy group", more related to resiliency grouping, and there seems to be some connection here too, but it was unclear. Please clarify.
- The problem scope is not well explained - is this intended to handle a small system with tens of servers, a mid-size system with hundreds, or a hyperscale system with tens of thousands of servers? How large can the servers be?
- The evaluation of scaling is poorly described. Mainly, it shows an evaluation with large numbers of groups and up to 10000 servers, but there are not enough details to understand what the "queries" measured are, and whether they adequately represent what might be seen in a production system,

- The target metric of the algorithms is unclear. Apparently, it incorporates capacity balancing, resilience, etc., but the connection is vague.
- I am not sure how the metric description would encompass notions of latency (response time) limits, or bandwidth limits for rebalancing, or limits on time delays for rebalancing (which would be important after failures, to restore resilience).
- Overall, it is unclear how precisely the algorithms work and how the components mesh together.
- How does the work connect with Deduplication? Or is it not intended to? (I realize that many dissertations are an aggregation of only loosely-related research papers, but, unless I missed something, this does not appear to be connected at all to deduplication, so I felt compelled to ask!)

Chapter 7: Conclusions
Overall this is fine.

Please contact me if there are any questions regarding my comments.

Sincerely,

Arif Merchant

Arif Merchant