Low-Overhead Byzantine Fault-Tolerant Storage*

James Hendricks, . . .

November 13, 2007

[Editorial comments are offset in brackets, like this.

Instructions: We have requested an outline to make your writeup easier (the outline isn't graded, but we'll have little sympathy if you neglect the outline and then fail the project writeup)—besides, you should have drafted an outline by this point any way. In particular, we'd like to ensure that (1) your writeup will be sufficient for you to pass and (2) you are not wasting time (yours in writing or ours in reading) with details not relevant to 15-712. Your outline is not a commitment. Please keep your outline under 2 pages (a half page is ideal)—see the half-page .txt version of this outline for an example.

Try to keep the writeup short—your writeup should be as short as possible, but no shorter. If it is less than five or six pages, your background, design, implementation, evaluation, related work, or analysis may be insufficient. If it is longer than fourteen or fifteen pages, some of the text is probably not relevant or concise (your audience may lose patience trying to read your report, which won't help your grade). Unnecessary detail is not impressive. There is no page limit, though—if you can describe an interesting systems solution, where it fits into the literature, and evaluation results in three pages, congratulations (warning: this is unlikely). Likewise, if you can keep your audience's interest for more than fourteen pages, congratulations (warning: this is even more unlikely). Please proofread and spellcheck. You are welcome (encouraged) to show the TA drafts of your writeup before the final report is due.

The TA based the following sample outline on a recent paper of his. You do not need to follow this outline—it is just what worked for him! Also, feel free to turn in a plain .txt file rather than a PDF.]

Abstract

Previous Byzantine fault-tolerant block storage protocols have either relied upon replication, which is inefficient for large blocks of data when tolerating multiple faults, or a combination of additional servers, extra computation, and versioned storage. Our protocol employs novel mechanisms to optimize for the common case and a new cryptographic primitive, homomorphic fingerprinting, to verify distributed erasure-coded data. Our protocol achieves throughput within 10% of the crash-tolerant protocol for writes and reads in failure-free runs when configured to tolerate up to 6 faulty servers and any number of faulty clients.

[Your abstract can be less detailed at this point.]

1 Introduction

1 page. This section will explain why Byzantine fault-tolerant storage is important and will note that prior solutions performed poorly because they didn't erasure-code data.

[This section should hint at why the problem is important and unsolved, and it should suggest your approach.]

2 Background

1.5 pages. This section will explain that (1) storage systems use redundancy for reliability, (2) erasure coding is needed for write performance, (3) making storage systems Byzantine fault-tolerant may not require any more servers or computation overheads (unlike Byzantine fault-tolerant replicated state machines), and (4) why prior Byzantine fault-tolerant storage solutions performed poorly (answer: mostly due to a lack of erasure coding).

[We will be looking for evidence somewhere in your writeup that the problem matters and has not already been solved. Reading related work and describing where your project fits in the literature is an important part of your grade.]

*This is a sample outline for the Fall 2007 Carnegie Mellon 15-712 class project. This outline is based on a recent SOSP paper [2].

3 Protocol design

2.5 pages. This section will describe the design of our protocol and how it differs from prior protocols.

[We will be looking for evidence somewhere in your writeup that your design choices are reasonable. Simple designs are always better than complicated designs—you will not impress us by providing too much detail!]

4 Implementation

2 pages. This section will describe the implementation of our protocol as well as our implementation of three competing protocols (PASIS-emulation, crash-tolerant replication-based, and crash-tolerant erasure-coded).

[This section could also describe the system simulator you start with and your improvements. You should tell us what you have done—there is no need to describe implementation work done outside of 15-712 in detail (just use a reference). We will be looking for evidence somewhere in your writeup that your implementation/simulation is convincingly realistic.]

We have implemented our protocol and the erasure-coded protocol, and the replication-based protocol is a special case of the erasure-coded protocol. We are debugging a seg fault in the PASIS-emulation protocol.

[If you provide a few details, we may be able to suggest approaches that will make your job easier. Please don't write too much!]

5 Evaluation

2.5 pages.

Experimental setup: 31 identical "server" nodes and 1 matching client on the same switch in a gigabit network (we have access to everything).

Experiment 1: Write throughput. Will generate a figure.

Experiment 2: Read throughput. Will generate a figure.

Experiment 3: Response time. Will generate 2 figures and 2 tables.

[We will be looking for some sort of experimental, simulation, or other validation that your approach would be worth pursuing in a real system. **Do not** exaggerate your results—so long as your design is reasonable, if you can only show a null hypothesis (e.g., your approach is no better than prior approaches), that's fine. Of course, it's easier for us to see that your design was reasonable if experimental results in a realistic environment say it performs well, but a surprising negative result is at least as interesting. Providing experimental evidence for or against your hypothesis (e.g., that your design is a good idea) is an important part of your grade. For most of the proposals, the TA was most concerned about this aspect of the writeup. You probably want to compare your approach against the current approach. You ought to show when your proposal works well but also when it fails (you won't fool anybody by only considering when your proposal is better; your readers will feel like you are trying to sneak one past them, which won't help your grade). Of course, how you evaluate your project depends on what your project is.

Once again, if you provide a few details, we may be able to suggest approaches that will make your job easier. Please don't write too much!]

Other section headings

[Here are a few other common sections. Of course, do what's appropriate for your writeup.]

6 Discussion

7 Related work

8 Future work

9 Summary or Conclusion

References

[1] [You will probably have between 10 and 40 references.]

[2] J. Hendricks, G. R. Ganger, and M. K. Reiter. Low-Overhead Byzantine Fault-Tolerant Storage. In *Proceedings of the 21st ACM Symposium on Principles of Operating Systems*, pages 73–85. ACM Press, 2007.