Getting Advice

(CS739 Fall '16 Midterm)

"No enemy is worse than bad advice" —Sophocles

The papers we read are forms of **advice**: lessons on what to do, and what not to do, when building or measuring systems. Your job on this exam: examine each piece of advice and decide whether it is good or bad advice, as well as a few other things. Read each question carefully and answer the best you can.

Good luck! And remember this OS-inspired advice: answer questions in **EASIEST-QUESTION-FIRST (EQF)** discipline. Following this advice will maximize the number of questions you finish before your time is up.

HELP ME, DR. BADADVICE. I KEEP HAVING DREAMS WHERE I'M FALLING OFF OF TALL BUILDINGS.
I THINK YOU SHOULD FOLLOW YOUR DREAMS.
WOW, THAT'S SOME GOOD ADVICE.
Cyanide and Happiness © Explosm.net

Name: ______________________

Student ID: ______________________

1. A person named J. Hamilton gives a lot of advice in his paper “On Designing and Deploying Internet Services.” One piece of advice he gives is this: “Support a big red switch.” What is the big red switch, and is having one in your system good advice or bad?

2. Jeff Dean gives a lot of advice about how to build systems, including numbers “everyone should know” (such as cache miss latency, mutex lock/unlock time, etc.) Why does he give the advice that everyone should know these numbers? Why are such numbers useful to system builders?

3. The RPC paper gives advice, of a sort, to send large packets by sending a smaller piece, waiting for an acknowledgement, then the next smaller piece, then waiting for an ack, etc. Is this good advice? When does this advice make sense, and when doesn’t it?

4. U-Net advocates (strongly) a position that networking be moved (mostly) to user level, yet despite this advice, some pieces of the U-net system remain in the kernel. Which pieces of networking functionality are placed in the kernel with U-net, and why?

5. Gray says the key to high availability (HA) is to "modularize the system". Why does he give us this advice, and what does it mean? Can you have HA without such modularization?

6. The Ding paper on "simple testing" shows the following diagram. What advice would you give system designers, based on this diagram? (put most important advice first)

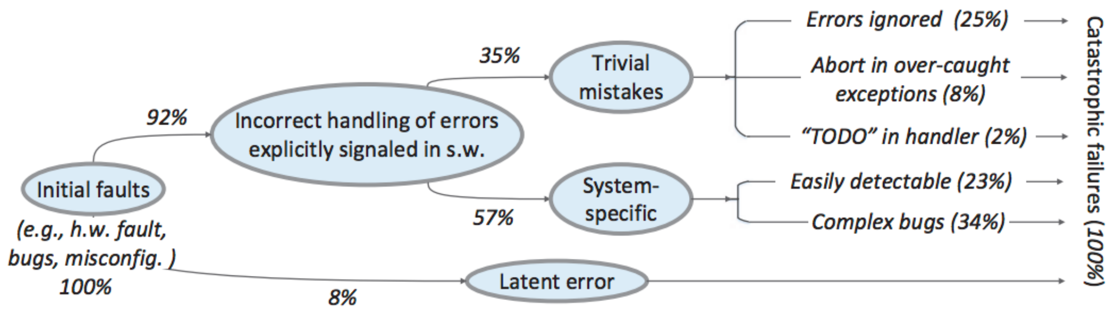
Initial faults
(e.g., h.w. fault, bugs, misconfig.)
100%
92%
Incorrect handling of errors explicitly signaled in s.w.
35%
Trivial mistakes
Errors ignored (25%)
Abort in over-caught exceptions (8%)
"TODO" in handler (2%)
57%
System-specific
Easily detectable (23%)
Complex bugs (34%)
8%
Latent error
Catastrophic failures (100%)

Figure 5: Break-down of all catastrophic failures by their error handling.

7. Schroeder concludes, in her study of disk failure, that disk failures do not take on a bathtub-like curve. What is a better depiction of the failure-rate curve she discovers? (draw this) Given this new curve, what advice would you give to large-scale storage system designers?

8. The ALICE paper discusses application-level protocols, and gives this piece of advice for Linux: when needed, make sure to call fsync() on the file of interest as well as on the parent directory. Why is this advice necessary? Does one need to always follow this advice to build a correct data-management application (such as the ones mentioned in the paper)?

9. In Sun's network file system, the basic protocol works particularly well for idempotent operations, so perhaps the advice here is to "always make operations idempotent". Despite this advice, not all NFS operations are idempotent. What types of operations are idempotent, and which are not? What kind of problems arise when non-idempotent operations are executed?

10. The two-phase commit algorithm we discussed in class advises that some logging steps need not be forced to disk immediately. Describe at least one such case, and explain why it is a good idea to follow this advice.

11. Remus describes an approach to high availability using “speculation”. They say that this form of speculation is needed to make their VMM-based HA system operate well. Why do they give this advice? First, describe what they mean by speculation, and then describe why it makes Remus work well.

12. The WiscKey paper advocates for the separation of keys and values in an LSM-based key-value store. When is this a good piece of advice, and when is it bad?

13. The Flash paper advises the use of “helper processes” in some cases. When are such processes needed? Is it ever a bad idea to use helper processes?

14. In both papers on latent-sector errors and disk corruptions, Bairavasundaram et al. note that such failures are **not** independent. What advice would you give designers of RAID storage arrays (or other replicated storage systems) assuming this is indeed the case?

15. A wise old systems person has been overheard giving the following piece of advice: “Always use vector clocks instead of Lamport clocks - they are strictly better!” Show a case where they indeed are better, and then decide if one should always follow this advice (and explain why).

16. In HA-NFS, the authors advise the reader to use different approaches in handling server, disk, and network failure. What was their specific advice on failure handling for these different types of failure? What do you think of this approach?

17. The Flash failure paper from CMU by Meza et al. suggests that “throttling” may help decrease SSD failure rates. What is throttling, and why might it be useful?