This position paper was prepared for and distributed at the 4th ACM SIGOPS European Workshop, Bologna, Italy, September 3-5, 1990, and published in Operating Systems Review 25, 1 (January, 1991), pages 81-82. Thanks to Jeremy Hylton for preparing this postable version, 10 March 1994.

Fault-Tolerance in Very Large Archival Systems

Jerome H. Saltzer
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

This position paper takes on a view of fault tolerance slightly distinct from the ones suggested in the workshop's call for participation. It does not come to any answers; instead it poses some new questions for discussion.

An interesting prospective application for computer systems is really-long-term archival storage of information, as has been traditionally performed by libraries, using books and serials. As costs of large-scale storage devices fall, librarians periodically reevaluate the question of whether or not to commit to digital storage any of the materials in their care. In the past there have been several concerns, but it has not been necessary to think very hard about them because the dominating concern has been that the cost of storage made the idea unworkable.

Recent advances in storage technology, and advances expected to appear over the next decade, promise to sweep aside the cost problem and finally bring the other concerns to the front.

One of the hard lessons that librarians have learned repeatedly from over-enthusiastic computer scientists during the last 30 years is that digital media have lifetimes far shorter than the archival times usually considered of interest in the library. Many libraries have taken into their care magnetic tapes, floppy disks, and similar computer-produced digital objects only to find a few years later--rarely even ten years--that although they may have carefully preserved the physical material itself according to the best recommendations of the time, there is no machine around capable of reading it. The old transport drives are no longer being manufactured, the last person who knew how to repair them has retired, and they couldn't be attached to any current computer system's I/O bus, anyway.

This lesson has been learned so well that librarians have acquired a properly healthy skepticism of all proposals to commit anything archival to digital form.

Since the information of interest resides in the bits rather than the physical medium, this skepticism strikes computer people as strange, because preserving the value of bits should be easier, technically, than preserving analog objects, such as books. From a systems point of view, the missing element in the design of a digital library system is a mechanism to copy information forward from old technology to new, in a "technology refresh cycle" measured in years. (In a way, this technology refresh system is analogous to the backup system that was identified decades ago as an essential component of systems that try to maintain persistent

disk storage. It takes more than an ad hoc warning to the system owner; a completely engineered system is needed.)

An interesting fault-tolerance question arises in working out the design of such a technology refresh system. Since a library is primarily an append-only storage system, with most objects once written never being modified, one might expect that fault tolerance for the archive could be effectively accomplished by simply relying on multiple, widely-separated storage sites. For example, one might implement a national library by placing three copies in three different sections of the country that are not likely to be simultaneously subject to disturbance by the same earthquake, windstorm, or civil disturbance. One would also encode the materials at each site sufficiently to provide reliable error detection, and assume that correction will be accomplished by retrieving a copy from one of the other two sites. But the addition of the technology refresh system to this distributed architecture may stress the fault-tolerant aspects in new ways.

The fault-tolerance problem has an extra edge on it because in a big, archival library, the first reference to an item may be 75 years after it is archived. This period until the next use is important, because if a fault corrupts the bits in an object, the next user will be the first to discover it. If the next use is within a few days, one of the duplicate copies at another site can be retrieved. If the next use is 75 years later, one must be careful that the duplicate copies are not by then similarly corrupted.

Looking ahead 75 years in the computer business is not easy, so we don't try to do it very often. How can we have confidence that someone reading the data then for the first time will find that it is actually readable? The refresh system is intended to make sure that the bits are not trapped in an obsolete medium. But that by itself does not insure against damage--the refresh operation may have happened ten times in the interim, and each time there are vulnerabilities to worry about. The refresh system would probably invoke the error detection machinery as it reads the data from the old medium, and replace any bad data with good data from another site. But errors seem likely to creep in while the data is in the hands of the refresh process itself. (It would seem wise to assume that any mechanism intended to copy 10 terabytes of data to a new medium is going to be make a few mistakes along the way.) It would appear that one would be well advised to include a fairly heavy duty form of forward ("end to end") error control. But if one provides this forward error control, are the multiple sites still necessary?

From a systems point of view, probably the most straightforward way to tackle these questions with some confidence in the answer is to divide up the various threats to integrity and provide independent counter-measures. Thus the multiple sites could be thought of as providing protection against large-scale physical disasters, while the forward error correction system provides protection against medium failures and copying mistakes. This division of responsibility (and admitted duplication of effort) also allows the optimizing parameters of each mechanism to be chosen independently, with a better chance of success of each of the fault-tolerance mechanisms.

On the other hand, a really heavy-duty form of forward error control would be to store three copies of the data at one site, with the technology refresh system organized to move the three copies to new technology at different times. But that approach might be just as well accomplished by doing the technology refresh independently on the single copy stored at

three different sites, thereby accomplishing protection from both threats at once. Is the result equivalent? Perhaps yes, if the three refresh cycles are completely independent. Almost certainly no, if they are somehow all done identically, for example with the refresh system crashing on disk overflow at exactly the same point in the refresh cycle at each site.

Thus, as usual, assuring independence of those failures that are inevitable ends up as one of the primary weapons in achieving fault tolerance.

———————————

Acknowledgement: The issues raised in this position paper were identified while the author was on a sabbatical assignment at the University of Cambridge, England, and at the Digital Equipment Corporation Systems Research Center, Palo Alto, California.