Low Cost Comparisons of File Copies*

Thomas Schwarz
Robert W. Bowdidge
Walter A. Burkhard

Computer Science and Engineering
University of California at San Diego

Abstract

The problem of maintaining consistency of replicas of large files has been addressed by Barbara, Feijoo and Garcia-Molina [1], Barbara and Lipton [2], Fuchs, Wu and Abraham [5] and Metzner [6]. Our model is essentially identical to that previously assumed. We present a scheme that provides all capabilities previously obtained as well as being able to detect and identify missing as well as extraneous pages. Our solution is an adaptation of Metzner's approach. We assume the existence of a signature function for individual pages. A supersignature is obtained as a power series in a primitive root within the Galois field with 2^n elements. The coefficients are the signatures of the individual pages. This scheme enables us to detect errors such as missing, altered or incorrectly placed pages with very high probability and to find an error diagnosis if discrepancies have been detected.

The problem solved in this paper is typified by the following situation. Consider a large database, replicas of which are situated at several sites. Each site keeps its own log file of updates and in order to insure consistency, these log files are compared periodically. The physical organization of the log file consists of pages.

Our model is the one usually adopted for this variety of problem. A file is organized as a sequence of pages and complete copies of the file are maintained at distinct sites throughout the network. We assume that few discrepancies occur and note that our scheme will not accommodate catastrophic failures. The cost of sending information through the network is high compared to the cost of calculations at an individual site. The order of pages is important and there might be missing pages besides faulty ones.

As usual, our solution avoids the costly and tedious task of bitwise comparison between the pages stored at different sites. Rather, we compare "signatures" of the individual pages by means of a supersignature calculated from the individual page signature. This allows conclusions that two copies are identical only with a previously chosen arbitrarily high probability. The cost of higher accuracy will be additional bits within the signature.

Our scheme is based on the calculation of the supersignature. We interpret page sig-

*This work was supported by the University of California MICRO program as well as the NCR Corporation, Dayton, OH

natures as elements of the Galois field $\mathcal{Z}_2^n$ of 2^n elements, but we disallow 0 as a value for a page signature. (We later propose variants that use more common data structures.) Let g be a fixed primitive root of $\mathcal{Z}_2^n$, i.e. a generator of the multiplicative group of $\mathcal{Z}_2^n$. The supersignature is then calculated as the weighted sum of the page signatures with the powers of g.

Definition 1 *Let $p_1, \ldots, p_n$ denote the page signatures. We define a supersignature H by*

$$H(p_1, \ldots p_n) := \sum_{\nu=1}^{n} g^{\nu} * p_{\nu}.$$

More generally, we define the supersignature of $p_k, \ldots, p_m$ to be

$$\sum_{\nu=k}^{m} g^{\nu} * p_{\nu}.$$

Note that the supersignature of a (contiguous) set of pages in a file does not only depend on the pages but also on the page number, that is, the placement of the pages in the file.

We have to show this supersignature is capable of detecting the existence of altered, missing, extraneous and misplaced pages and we present an algorithm that given a small number of discrepancies of the replicas of the file finds a proper error diagnosis.

Effectiveness

There are several considerations regarding the efficiency of a signature. The computation of the signature must be feasible. Error detection is another measure of efficiency. We assume that the supersignature is to be a good hash function on the set of all pages, that is, the probability of attaining any particular value is identical for all possible values. In this sense it can be shown that H is good if the page signature is good.

We can also give probabilities that one error of a particular kind can be detected. The probability that one altered page is detected is equal to the probability that a page signature does not detect an error, which is $1/2^n$. A single missing or extraneous page will be detected because the total number of pages does not agree. Even without this information the supersignatures will not agree if the page signature of the missing page is not zero (which can happen only with probability $1/2^n$. An inversion (that is a case where two pages have exchanged places in the file) will be noticed with probability $\approx 1/2^{n-1}$. A jump, where one page shows up later in one copy than in the other, will be detected with the same probability. (In all this we made the silent assumption that page signatures are independently distributed.)

Sometimes, page signatures are selected because of their error detecting probabilities. This however tends to generate good hash functions. A simple counting argument shows that error detection is impossible for supersignatures (at least if the number of pages in a file is not limited.)

Some Properties

Subsupersignatures, that is, supersignatures of a contiguous subset of pages in a file, are dependent on the placement in the file. That is, exactly the same set of pages will yield different supersignatures according to the pagenumber the first page of this subset has. However, one will be the g^k multiple of the other, where k is the difference of the respective page numbers.

We also have additivity. If we know the supersignature of a (contiguous) set of pages and the supersignature of the first (or second) half of this set then a simple subtraction will yield the supersignature of the other half. This property will cut the amount of bits transmitted during the diagnosis of a discrepancy by half.

A File Comparison Algorithm

We can now use a variant of Metzner's strategy to compare two files.

In overview, Site I (the initiator site) requests the number of pages in Site's II copy of the file as well as the supersignature and then calculates its own data from its local copy. If the data do not agree, the file is split in halves and each copy's data are compared. But not only are these parts of the file compared, but it is also tested whether the lacking consistency is due to misalignment caused by lacking pages in either copy of the file. This is done by conceptually shifting the parts of Site's I copy to the left and to the right.

We will work with pairs (H, n) consisting of a supersignature of a contiguous set of pages and the number of pages. Two such pairs are the same exactly when both components agree. In the course of error detection our increased knowledge of both file copies in terms of a breaking up of the supersignatures in those covering smaller numbers of pages will be represented by a sequence of those pairs:

$$((H_1^{(i)}, n_1^{(i)}), \dots (H_k^{(i)}, n_k^{(i)}))$$

the meaning of which is that the first n_1 pages of the copy at site i have supersignature $H_1^{(i)}$, and so on. We will have to define two operations on pairs: Leftshift and Rightshift. The first one will correspond to conceptually destroying a page to the left of the area of pages in question thus simulating a missing page in the other site's copy or a superfluous one in our copy whereas the other operation correspond to inserting a dummy page to the left thus simulating a missing page in our copy or a superfluous one in the other copy. These operations are not always defined if the set of pages covered does include the first or the last page.

Assume that (H', n') covers the pages $\{p_m, p_{m+1}, \dots p_{m+n}\}$. Conceptually destroying page p_1 now results in having to replace H' by the supersignature of the pages $\{p_{m+1}, \dots, p_{m+n}, p_{m+n+1}\}$. This can be calculated by $H^{\text{new}} := g^{-1}H' - g^m h(p_m) + g^{m+n} h(p_{m+n})$. A similar formula can easily be derived for the Rightshift.

The scheme proper now starts out by Site I, which is the calculation intensive site, requesting the number of pages and the supersignature of site's II copy of the file and comparing it with its own. Also, the number of errors allowed is set to F, a previously agreed upon number.

If in general we have reached a state in which we want to compare two sequences of pairs

$$((H_\nu^{(1)}, n_\nu^{(1)})_{\nu=1,\dots,k}) \qquad ((H_\nu^{(2)}, n_\nu^{(2)})_{\nu=1,\dots,k})$$

we compare the corresponding pairs sequentially in the manner described below. If during a comparison of a pair $(H^{(1)}, n^{(1)})$ with $(H^{(2)}, n^{(2)})$ it turns out that $n^{(i)} = 0$ the corresponding set of pages at the other side is labelled superfluous. If $n^{(1)} = 1 = n^{(2)}$ and the signatures are not the same these pages are labelled as different. Note that we actually are not diagnosing inversions as such, this is left to the error recovery routines. In these cases, the pairs are eliminated from further consideration. In all other cases, inequality of pairs is diagnosed. But this is first tried to be resolved by Left- and Rightshifting of the pair corresponding to site I by up to F (the number of errors allowed) shifts in either direction. When through this maneuver or straight from the beginning equality of pairs has been found the part of both sequences that is to the left of the pair in question is separated from the rest and the error allowance F decreased by the number of shifts necessary. In the same move the equal pairs are eliminated from further consideration as they are deemed to be equal. If equality cannot be achieved, the set of pages in question is divided in half in such a way that the highest power of 2 strictly smaller than $n^{(2)}$ is the number of pages in the first half. Site I has to request the corresponding supersignature from site II and initiate the appropriate calculation for its part. The supersignature of the other half can be calculated from the old and the just obtained one. For every splitting of a pair, the error allowance F is decreased by one, as we just proved the existence of one more mistake, After each comparison, we end up with a set of sequences with a total of at most F pairs that are dealt with recursively. The algorithm stops when F becomes negative, i.e. in failure, or when there is nothing left to be considered.

Example

Assume, that site I contains the file with page signatures

Site I: $l, m, n, o, p, q, r, s, t, u, v, w, x, y, z$

and that site II contains the file with page signatures

Site II: $l, m, o, p, q, r, s, t, u, v, w, x, y, z$

Note that the page with signature n at site II is missing. In the initial step, site I notices that the number of pages differ. It nevertheless requests

$$H_2(l, m, n, o, p, q, r, s, t, u, v, w, x, y, z).$$

Then site I asks for and compares

$$H_1(l, m, n, o, p, q, r, s) \neq H_2(l, m, o, p, q, r, s, t)$$

and calculates H_2 and compares

$$H_1(u, v, w, x, y, z) = H_2(u, v, w, x, y, z)$$

because only values of H of the same number of arguments Should be compared. At this point site I knows that there is at least one error in

I: $l, m, n, o, p, q, r, s, t$ II: l, m, o, p, q, r, s, t

(which incidentally also follows from the fact that the number of pages do not agree.) In the next round site I compares

$$H_1(l, m, n, o) \neq H_2(l, m, o, p)$$

and calculates $H_1(q, r, s, t)$ which is compared to $H_2(p, q, r, s)$ and $H_2(q, r, s, t)$. Site I finds that file (q,r,s,t) at I is identical to (q,r,s,t) at site II and hence site I only has to compare files

I: l, m, n, o, p and II: l, m, o, p

After comparing

$$H_1(l, m) = H_2(l, m);$$
$$H_1(n, o) \neq H_2(o, p);$$
$$H_1(o, p) = H_2(o, p)$$

site I can now conclude that its third page is not to be found in the second site's record.

Variations of the Algorithm

The algorithm is biased towards the assumption that errors are due to missing pages. For a small number of actual errors this will not pose a problem, the algorithm will give the correct error diagnosis with high probability. If however one tries to use this algorithm under more bizarre circumstances, obvious improvements can be made.

In rare cases the algorithm might fail to find the right diagnosis because supersignatures of subsets of pages coincide even though the subsets do not and the algorithm is led astray. It will never make a wrong diagnosis unless several page signatures of different pages agree. If several explanations exist, it might however not give the most probable one.

To guarantee a diagnosis the following modifications are made. We impose an upper error bound E for the number of altered pages and another, F, for the number of missing or superfluous pages. In this context, it should be noted that a jump or an inversion counts as one missing and one superfluous page. The number of shift operations is then restricted to F. If the algorithm fails, rerun with increasing values of $F = 1, \ldots$ will lead to a correct answer.

In general, the algorithm will be able to detect a greater number of missing pages than the amount of F, even if these missing pages are contiguous.

Alternative Signature Spaces

In the preceeding discussion we equipped the signature space, i.e. the space of all possible signatures, with the algebraic structure of a Galois field. (This is the only algebraic structure satisfying our needs which can be defined on a set of this size.) To make our scheme computationally accessible, we have to multiply powers of g. This task has been solved by the dual base algorithm due to Elwyn Berlekamp (v. [3, 7]).

In practice, other implementations of arithmetic on a slightly altered signature space might be preferable. Though the space of all bit strings of a given length is the best signature space for generation purposes, the implementation of Berlekamp's algorithm or any other implementation of multiplication in a Galois field might be considered tedious.

Our results remain valid if the signature space is the set of integers modulo a given prime (usually the largest prime smaller than the largest unsigned integer directly representable on the machine) or a cartesian product of the integers modulo different primes. The arithmetic operations in the latter case are defined componentwise. The generating element is a vector consisting of primitive elements modulo the i^{th} prime in the i^{th} coordinate. Presumably, page signatures are generated as a string of bits. If we interpret these strings as unsigned integers and take the remainder modulo these primes we can represent $\Pi_{i=1}^{N} p_i$ different signatures (where $p_i = 2^w - a_i$ are the primes and w denotes the word length.) This scheme is known as Chinese Remaindering. The values with i^{th}*component* smaller than a_i will be attained with double the probability than the larger ones. This is not a big disadvantage, as these values will only be attained with a very small probability and as to achieve the same guaranteed bounds of the original scheme we have to make signatures N bits longer. Alternatively, we can alter the page signature generation scheme. The big advantage of this alternation lies in the use of commonly used operations that are optimized in any machine.

Implementation

A version of this algorithm has been coded and tested ([4]). The signature space consisted of vectors of dimension five the coordinates of which were remainders modulo the five largest primes smaller than the maximal representable unsigned integer. To test the algorithm a signature file was randomly generated and transmission of the original file to another site simulated by capturing transmission errors by its effects on the signature file. Four types of errors were considered: Altered pages ($\frac{15}{21}$), lost pages ($\frac{2}{21}$), jumped pages ($\frac{3}{21}$) and doubled pages ($\frac{1}{21}$), where the numbers are the probabilities for this kind of error to occur. To measure the costs the two resulting signature files were subjected to our comparison algorithm and the number of multiplication and remote requests recorded with the total number of errors as a parameter. (See figures 1 and 2.)

We also compare our algorithm with the naive one, which transfers the file and compares it bit by bit, measuring the time required by each one. We present both experimental and analytical results for the two comparison schemes. These are typical of computation environments including Sun 3/50 sites connected with 1 Mb/sec networking. Our simulation times were obtained using Sun 3/50.

Assuming a file size of 10000 pages of 1K byte each, the naive approach causes 81920000 bits to be sent. Our algorithm will cause $\log_2(10^4)$ remote requests for supersignatures in the presence of one error and hence at most $1 + E\log_2(10^4)$ for E errors resulting in network traffic of 161392 bits assuming 100 errors and an integer size of 2 bytes or around 500 times less traffic.

The performance comparison is represented in the table below. The entries in the comparison row measure the time required to do the necessary computatinal comparisons. For the naive algorithm, the result depends on the file size and is otherwise constant. For our algorithm the result is the average for 20 experiments each consisting of 200 samples where the number of errors was fixed in each experiment. The average comparison time increased from 21 to 41 seconds as we varied the number of errors from 1 to 20.

The entries in the transmission row measure the time required to transmit the necessary information. In the naive algorithm the file itself is transmitted while in our algorithm an inconsequential amount of data is transmitted.

	Our alg.	Naive alg.
comparisons	29 sec	54 sec
transmission	≈ 0 sec	80 sec

These times assume that it is not necessary to compute the signature as they can be maintained incrementally. In any case, the time to compute the supersignature would be approximately the time required by the byte–by–byte comparison.

Appendix: Mathematical Results

In this section, we assume that the signatures are represented as elements of $\mathcal{Z}_2{}^n$, but that the value 0 is not a permissible signature.

Lemma 1 *Assume that $p_\nu = \hat{p}_{\nu+c}$ where $p_k, \dots p_l, \hat{p}_{k+c}, \dots, \hat{p}_{l+c} \in P$ (the set of page signatures and $c =$ constant.*
Then

$$g^c \cdot H\left((p_k, \dots, p_l)\right) = H\left((\hat{p}_{k+c}, \dots, \hat{p}_{l+c})\right).$$

Proof:

$$H\left((\hat{p}_{k+c}, \dots, \hat{p}_{l+c})\right) = \sum_{\nu=k+c}^{l+c} g^\nu \cdot h(\hat{p}_\nu) = \sum_{\nu=k}^{l} g^{\nu+c} \cdot h(\hat{p}_{\nu+c}) = g^c \cdot \sum_{\nu=k}^{l} g^\nu h(p_\nu) = g^c \cdot H\left((p_k, \dots, p_l)\right) \quad \blacksquare$$

The following observation shows that we can calculate the supersignature of the first (resp. last) half of a set of contiguous pages from the other half's and the whole set's supersignatures.

Lemma 2

$$H((p_l, p_{l+1}, \dots, p_k)) = H((p_l, p_{l+1}, \dots, p_{m-1})) + H((p_m, p_{m+1}, \dots, p_k))$$

Lemma 3 *An inversion will be detected with probability approximately 2^{-n}.*

Proof: We calculate the probability that an inversion (i.e. an exchange) of two pages will not be noticed by H. Let the first page have signature s_1 and page number i at the first site and let the second page have signature s_2 and page number j at the first site and assume that these page numbers are reversed at the second site. Then H will not detect an inversion iff

$$g^i s_1 + g^j s_2 = g^j s_1 + g^i s_2.$$

This is equivalent to

$$(g^i - g^j)s_1 = (g^i - g^j)s_2,$$

i.e., iff

$$s_1 = s_2 \text{ or } g^i = g^j.$$

The later condition is equivalent to i being congruent modulo $2^n - 1$ to j. Therefore the probability that H will not detect an inversion is only slightly higher than 2^{-n}. $\blacksquare$

Lemma 4 *A jump will be detected with probability 2^{-n}.*

Proof: Consider:

$$p_1, p_2, \dots p_{k-1}, p_k, p_{k+1}, \dots p_j, \dots p_m$$

at site I and

$$p_1, p_2 \dots p_{k-1}, p_{k+1}, \dots p_j, p_k, p_{j+1}, \dots p_m$$

Note that page p_k "jumped". Then

$$H_I - H_{II} = g^k h(p_k) + g\left(\sum_{\mu=k+1}^{j-1} g^{\mu-1} h(p_\mu)\right) + g^j h(p_j) - g^k h(p_j) - \sum_{\mu=k+1}^{j-1} g^{\mu-1} h(p_\mu) - g^j h(p_k)$$

$$= (g-1)\left(\sum_{\mu=k+1}^{j-1} g^{\mu-1} h(p_\mu)\right) + (g^k - g^j)(h(p_k) + h(p_j))$$

The probability that H_I equals H_{II} is therefore 2^{-n}. $\blacksquare$

References

[1] D. Barbara, B. Feijoo and H. Garcia-Molina: Exploiting Symmetries for Low-Cost Comparison of File Copies; *Proc. International Conference on Distributed Computing Systems,* San Jose, June 1988

[2] Daniel Barbara and Richard J. Lipton: A class of Randomized Strategies for Low-Cost Comparison of File Copies; Technical Report Princeton CS - TR 176-88 ; 9/88

[3] Elwyn R. Berlekamp: Algebraic Coding Theory; *McGraw-Hill Book Company* New York 1968

[4] R. W. Bowdidge, W. A. Burkhard, T. J. E. Schwarz, Technical Report UCSD , in statu nascendi

[5] W. K. Fuchs, K. Wu and J. Abraham: Low-Cost Comparison and Diagnosis of Large Remotely Located Files; *Fifth Symposium on Reliability in Distributed Software and Database Systems,* January 1986, 67-73

[6] J. Metzner: A Parity Structure for Large Remotely Located Data Files; *IEEE Transactions on Computers* Vol C - 32, No. 8, 1983

[7] Robert J. McEliece: Finite Fields for Computer Scientists and Engineers; *Kluwer Academic Publishers* ; Boston, Dordrecht, Lancaster 1987