Estimating Duplication by Content-based Sampling

Fei Xie Michael Condict Sandip Shete
{fei.xie, michael.condict, sandip.shete}@netapp.com
Advanced Technology Group, NetApp Inc.

Abstract

We define a new technique for accurately estimating the amount of duplication in a storage volume from a small sample and we analyze its performance and accuracy. The estimate is useful for determining whether it is worthwhile to incur the overhead of deduplication. The technique works by scanning the fingerprints of every block in the volume, but only including in the sample a single copy of each fingerprint that passes a filter. The selectivity of the filter is repeatedly increased while reading the fingerprints, to produce the target sample size. We show that the required sample size for a reasonable accuracy is small and independent of the size of the volume. In addition, we define and analyze an on-line technique that, once an initial scan of all fingerprints has been performed, efficiently maintains an up-to-date estimate of the duplication as the file system is modified. Experiments with various real data sets show that the accuracy is as predicted by theory. We also prototyped the proposed technique in an enterprise storage system and measured the performance overhead using the IOzone micro-benchmark.

1 Introduction

Deduplication detects and removes duplicate data blocks (blocks at different locations that have the same contents) from a storage system. In a system implementing perfect deduplication, only one copy of duplicate data blocks is stored, but in such a way that the user's view of the system remains unchanged. A. El-Shimi et al. [21] provide a nice overview of the recent research work in the area of data deduplication.

The benefit of deduplication in a primary storage system varies for different workloads. For certain workloads that have a low level of duplication, one would turn off the deduplication feature to avoid its effect on I/O performance and to avoid the metadata overhead of deduplication. It is desirable to have an efficient and effective deduplication estimator to allow customers to quickly estimate the deduplication benefit on their primary data sets before they turn on deduplication, and to allow the storage system to prioritize the scheduling of deduplication tasks for different data sets.

Existing deduplication estimators are either not fast enough or not accurate enough. A simple but intrusive and time-consuming way to discover the benefit of deduplication is to actually turn on deduplication. If the benefit is not satisfactory, deduplication can be reverted. Alternatively, one could roughly estimate the potential benefit of deduplication based on the type of workload. This approach often does not produce accurate estimates, since it does not look at the content of data.

Taking the content into account, one could attempt to estimate the level of duplication by reading a small random sample of the data set, and calculating the amount of duplication in it. This is much harder than it sounds, because of the large error in estimating the true number of occurrences of an item that only occurs once or twice in the sample. Furthermore, it has been proven that for any random-sampling-based estimation function, there are block-frequency distributions that cause it to be very inaccurate, unless the sample percentage is very large fraction of the data [1].

We defined and implemented an accurate and lightweight deduplication-estimation technique for a primary storage system. At a very high level, the technique samples the blocks based on the block fingerprint value, and only selects the fingerprints that satisfy some predicate on their value (i.e., a filter). That is, any two blocks with the same fingerprint are either both included in the sample or both excluded from the sample. This is why the sample is said to be content-based.

The remainder of this paper consists of a comparison to previous work (Section 2), an analysis of the algorithm (Section 3), a discussion of the design and implementation of the system (Section 4), a performance study (Section 5), and our conclusions (Section 6).

2 Related Work

Many commercial storage vendors provide deduplication estimators to allow customers to estimate potential space savings. A popular approach is to use rule-of-

thumb estimation methods which involve looking at metadata information like the type of data set, the frequency that data is changed, annual data growth rate, data retention, etc., which tend to influence deduplication ratios [11]. Many commercial estimators [9], [10] have adopted this method. Note that these estimators do not access the actual data in order to calculate the estimates, and so, can sometimes be extremely inaccurate.

The problem of estimating the duplication in a data set can be thought of as estimating the number of distinct block values in the set, given that the total number of blocks in use is known. This means that solutions to the latter problem can be applied to the former.

Distinct elements estimation is a well-studied problem, and frequently appears in literature concerning data-streaming algorithms, statistics, and databases. P.B. Gibbons [8] looks at many previous approaches to distinct value estimation and the difficulties with them. In the database world, the early literature has extensively studied sampling techniques, which involve gathering a uniform random sample of the data, and using it to approximately answer distinct-value queries on relational databases [5, 6, 7]. Although these estimators use sophisticated techniques to handle various input distributions, they are all unable to guarantee good accuracy for their estimates [1, 4]. Charikar et al. [1] proved this formally, establishing a strong negative result, namely that no estimator can guarantee a small error for all possible input distributions unless it examines a large fraction of the input data. Raskhodnikova et al. [2] and Valiant et al. [3] further provided near-linear and sub-linear lower bounds, respectively, on the sample size required for the estimate. The conclusion is that, in order to ensure high-accuracy, distribution-independent estimates, it is necessary to examine almost the entire data set.

Estimation algorithms that require scanning all the data once are referred to as single-pass algorithms. Flajolet and Martin, in their seminal work [12], presented the first single-pass algorithm for distinct values estimation in a large collection of data using small limited storage. Their probabilistic counting algorithm uses hash functions to map set of values to bitmap vectors, such that each distinct value maps to the i^{th} bit in the vector with $2^{-(i+1)}$ probability. Alon et al. [13] further build upon this work and proposed more practical hash functions, space bounds and provable error guarantees on their estimates. This line of research continues with more space/time efficient algorithms and better estimates of distinct values [14, 15]. Some similar approaches use adaptive sampling, which continuously maintain a bounded-size up-to-date sample of distinct values for the purpose of providing a very quick estimate of the cardinality of the data set [17, 18, 26]. Our sampling technique is in many respects similar to these.

D. Harnik et al. [19] are the first in the area of storage deduplication to provide a provably accurate two-phase algorithm for a one-time estimate of deduplication ratios, using very low storage space. Our technique differs from theirs in that, after a single pass over existing data for the initial estimate, it uses an adaptive technique to incrementally maintain an up-to-date estimate that takes into account any changes to the data.

3 Theory

Consider a data set consisting of a group of data blocks (of fixed size or variable size) with possible duplicates. We are interested in estimating the percentage of space that can be saved by deduplication. Thus, we define the deduplication ratio R as follows.

$$R = \frac{Size\ in\ bytes\ of\ distinct\ blocks\ in\ data\ set}{Size\ in\ bytes\ of\ the\ data\ set}$$

We assume a hash function that generates a fingerprint for each data block. The proposed *content-based sampling* applies a modulo-based filter to all the block fingerprints of a data set. A block fingerprint passes the filter and is added to the sample iff:

$$Fingerprint\ \mathbf{Mod}\ M = X$$

Where the divisor M is an integer greater than 1, and the remainder X is an integer between 0 and M - 1. Throughout this paper, we refer to M as the *filter divisor*. The idea is to split the fingerprint space into M partitions, and to use one of the partitions in the estimate. More specifically, the total size of the distinct (deduplicated) blocks in the sample is used to estimate the total size of the distinct blocks in the entire data set.

Assume there are K different block sizes in the data set. The sample can be partitioned into K groups of identical-sized blocks. Assume the ith ($i = 1 \ldots K$) block group in the sample has n_i distinct blocks of size s_i. Let N_i denote the total number of distinct blocks of size s_i in the data set. Let S be the size in byte of the distinct blocks in the data set. The estimate of S, denoted by S^*, is defined as:

$$S^* = M \cdot \sum_{i=1}^{K} n_i \cdot s_i.$$

The deduplication ratio can be estimated as $S^*/S_{\text{data\_set}}$, where $S_{\text{data\_set}}$ is the size of the data set before deduplication, which is known before the. In the case of fixed-size blocks, we can ignore the block size and count only the number of distinct blocks in the sample. Figure 1 illustrates the idea of content-based sampling.

The main theoretical result of this work is the relationship between the filter divisor and the accuracy of the estimate. Define the relative error of the estimate as

$$err = (S - S^*)/S$$

We show that if the fingerprinting algorithm has good uniformity and negligible collision probability, the relative error follows a normal distribution with zero mean (i.e., the estimate is unbiased) and known variance.

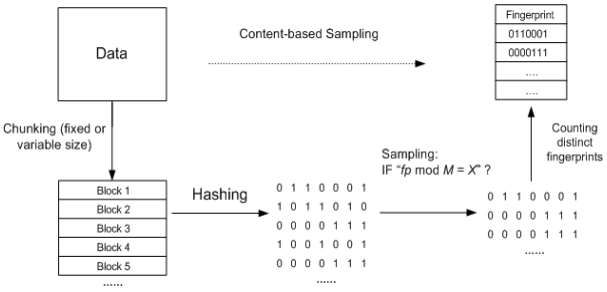

Figure 1. Illustration of the Content-based sampling

We assume a storage system that maintains a relatively strong fingerprint for data blocks, so that we can ignore the impact of collisions. A previous work in the networking domain [16] takes collisions into account in the estimate. As shown in that work, the expected error introduced by collisions (which always causes an underestimate) is computable from the size of the uncorrected estimate S^*. Thus, we could apply these results to correct for collisions, if necessary.

Theorem 1. Assume the fingerprinting has the uniformity property and negligible collision probability. For large N_i (i = 1,.., K), *err* has a normal distribution with zero as the mean and $(M-1)\cdot \bar{s}/S$ as the variance, where $\bar{s}$ is defined as $\bar{s} = \sum_{i=1}^{K} s_i \cdot (N_i \cdot s_i)/S$.

The proof of Theorem 1 is in Appendix A. The *α-β* accuracy of the estimate is defined as:

Definition of *α-β* accuracy. Given α and β ($\alpha, \beta \in [0, 1]$), the relative error *err* satisfies the following condition.

$$Prob(|err| \geq \alpha) \leq 1 - \beta$$

The relationship between M and the *α-β* accuracy is described in the following theorem.

Theorem 2. M satisfies the *α-β* accuracy if:

$$M \leq \frac{\alpha^2 \cdot S}{2\cdot(erf^{-1}(\beta))^2 \cdot \bar{s}} + 1 \qquad (1)$$

where $erf^{-1}()$ is the inverse of the Gauss error function and $\bar{s}$ is defined in Theorem 1 (proof in Appendix B).

To have the smallest possible sample size, one would choose the largest M that satisfies a given accuracy requirement. However, S and $\bar{s}$ are not known before the estimation. In practice, we could address this problem as follows. Variable-size chunking algorithms (e.g., [24]) typically have a known average block size, which can be used to approximate $\bar{s}$. Also, $\bar{s}$ is known for the fixed-size blocks case. Rewrite inequality (1) as:

$$\frac{S}{M-1} \geq \frac{2\cdot(erf^{-1}(\beta))^2 \cdot \bar{s}}{\alpha^2} \qquad (2)$$

The left side of (2) could be approximated by the total size of distinct blocks in sample (distinct block count in the fixed-size blocks case), which is countable during the sampling. The minimum "distinct sample size" that satisfies (2) is called the *target sample size*. Table 1 gives some α, β values and the corresponding target sample size in number of blocks. As long as there are enough distinct blocks in the sample, we can increase the selectiveness of the filter to reduce sample size in the following *Adaptive Sampling* Approach.

During the sampling process, the ratio of the distinct block count in sample to the *target sample size* is periodically monitored. If the ratio is greater than 2, we find the largest power of two that is less than or equal to the ratio (denoted as *f*). M and X are updated as follows:

Step 1: $X = X + M \cdot rand(f)$
Step 2: $M = M \cdot f$

The function rand(*f*) generates a random integer between 0 and $f - 1$. This allows us to randomly choose a fingerprint partition while we aggressively divide the fingerprint space. Finally, we remove the unqualified blocks from the existing sample, and continue the sampling with the new, more restrictive filter.

TABLE 1. Target sample size vs *α-β* accuracy

Target sample size	α	β
270	0.1	0.9
1843	0.06	0.99
12030	0.03	0.999
1513670	0.01	0.9999

4 Design and Implementation

We chose an enterprise-class network-attached storage system as the reference system in which to implement the estimation technique. The system uses a log-structured file system [22] with 4KB blocks. Individual data blocks of a file can be identified by a file handle together with an offset within the file (called a file block number). Our implementation estimates the number of unique data blocks in a volume. Ignoring the initialization delay due to the one-time full-volume scan, an up-to-date estimate can be returned on-demand with-

in a couple of minutes, no matter how large the volume is. A large part of the recurring maintenance runs in the background, in a non-intrusive fashion.

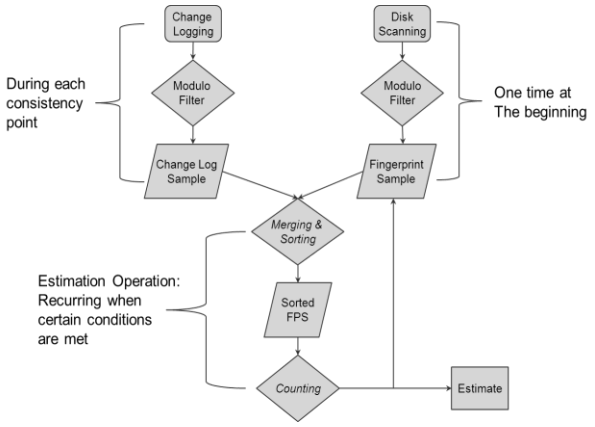

Figure 2. System Components

The major modules built into the reference storage system are depicted in Figure 2. The ***change logging*** is a software module that samples *data blocks* during the consistency point. Metadata blocks are not sampled. The storage system computes a 64-bit variant of the Adler checksum [23] for each block as the RAID checksum. This checksum, available at the time of a consistency point, is used as the fingerprint. Existing blocks in the volume are sampled by a ***scanner*** module. The sample is stored in a fingerprint sample file (FPS). The FPS contains a header and a sequence of entries (20 bytes per entry) in the format of {file handle, file block number, fingerprint}.

The ***estimation operation*** merges the sample from change logging to the FPS, and updates the estimate accordingly. File swapping is used in the merging, so that the ongoing change logging process is not affected. After merging, we count the distinct blocks by sorting the FPS. We remove stale entries (i.e., blocks removed or over-written) before counting. This is done by comparing the fingerprint in the entry with the fingerprint of the real block. We maintain a stale inode cache during the validation, to reduce the number of unnecessary block-read attempts.

Adaptive sampling is triggered at the end of the estimation operation. Once the filter divisor increases, the change logging produces a smaller sample. The FPS is shrunk as well. The initial value of M can be set by the user. The initial value of X is chosen randomly.

The estimation operation is triggered if we have enough data in the change logs or there is file deletion in the volume (i.e., volume size decrease). These conditions are checked periodically. This ensures minimum impact to read-intensive workloads.

5 Performance Study

We studied the accuracy of our estimate using real-world data sets (see TABLE 2). Given a data set, we compared the empirical error's standard deviation and the theoretical ones, for various values of M. There were 1000 data points for each empirical statistics, generated by varying the remainder X from 0 to 999. We tested estimations over both 4KB fixed-size blocks, and variable-size blocks [24]. The variable block size is between 2KB and 8KB. The true deduplication ratio was obtained as follows. In the case of variable-size blocks, we trust the results of deduplicating over the MD5 hash values of the blocks. For fixed-size blocks, we deduplicated the data set in a NetApp® system.

TABLE 2. Information of the data sets

Names	*Size*	*Dedupe Ratio*	*Description*
Corp. Web	1.5TB	~50%	Corporate web directory
Debian	260GB	~60%	2-month Debian build
Sharepoint	29GB	~18%	Corporate Sharepoint

We consistently saw good matches between the empirical results and the theoretical results, for both the variables-size and fixed-size cases (see Figure 3 for details). For the sake of space, we only report selected results in this paper.

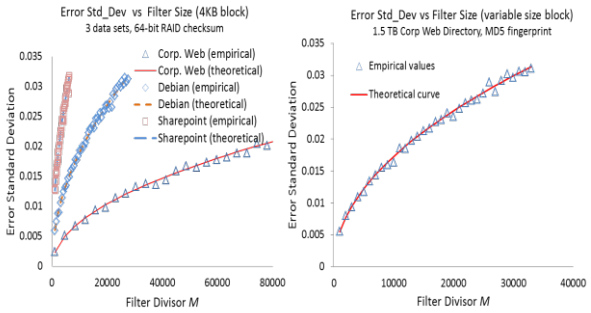

Figure 3. Accuracy test result

The evaluations of the prototype were done using a NetApp FAS 3070 storage system running Data ONTAP® 8.1 [25]. IOzone [20] was chosen as the synthetic I/O trace generator, since it can generate traces with duplicated content. The storage system exported a NFS v3 volume to the trace-generating client.

There are two major types of test, namely *64KB sequential write* and *64KB sequential read*. All the tests were set to 50% inter-file duplication and 0% intra-file duplication. Five files are accessed in parallel in the tests, which saturated a 1GB network link. The deduplication ratio of the synthetic data set is 0.6. Every 4

minutes, the system checked whether the estimation operation should be triggered. Every 5 seconds, the storage system sampled the CPU usage, number of 64KB I/O per second (IPOS), disk read rate, and disk write rate. A single test run lasted for about 80 minutes. There were 15 test runs for a single setting.

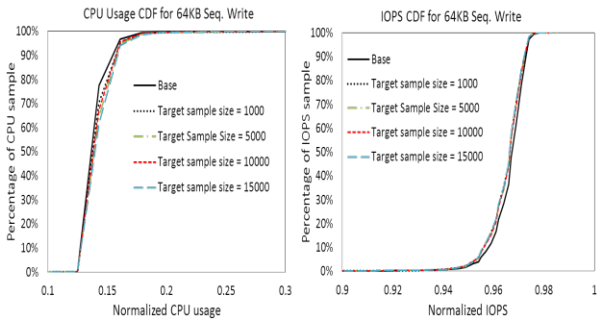

Figure 4. CDF for normalized CPU usage and IOPS

We first look at the results for 64KB sequential writes. We only report the cumulative distribution function (CDF) curves of CPU usage and IPOS in Figure 4. As expected, the estimation test consumes more CPU and has less IOPS, compared to the base case. Besides these results, our data shows that the average changes in CPU usage, IOPS, and disk reads are less than 0.5% for all the tested target sample sizes. The average disk read rate increases from 1% to 4% as the target sample size increases from 1000 to 15000. The additional disk reads are mainly contributed by the random disk access from the counting module.

Figure 5. Latency histogram for both read and write tests

There is no significant impact to the system in the sequential read test. This is because the estimation code has negligible overhead for a read-only workload. The CDF results are not reported due to space limitations. Figure 5 plots the client-side latency histogram reported by IOzone. This plot shows that estimation's impact to the I/O latency is also negligible.

We also studied the estimation accuracy of our adaptive sampling in the 64KB sequential write case. Figure 6 plots the results in one test run. As the volume size grows, the filter divisor is doubled while the corresponding distinct block count in the sample floats between 2500 and 1000. *M* was initially set to 4096. The error is high at the beginning due to the small sample size, and remains below 5% later.

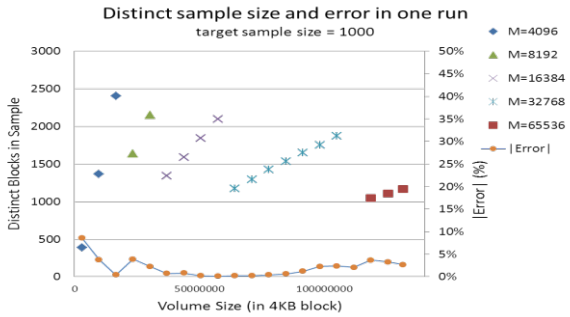

Figure 6. Adaptive sampling in one experiment run.

6 Concluding Remarks

The main contribution of this work is a method that estimates duplication in a storage system with statistically guaranteed accuracy using a single scan of the data set. It is also notable for requiring only a small, fixed amount of memory resources for a given level of accuracy, independent of the size of the volume. This makes it amenable to efficient in-line use and to maintain an accurate estimate in the face of a rapidly changing data set, which allows storage users to better assess the utility of data deduplication. We implemented the technique as a practical enhancement to a commercial storage system, and confirmed that the accuracy was within the statistically expected range for a variety of real-world data sets. The performance impact of our technique was found to be less than a few percent.

7 References

[1] M. Charikar, et al. Towards Estimation Error Guarantees for Distinct Values. Proceedings of the 19th ACM Symposium on Principles of Database Systems. ACM, New York, 2000.

[2] S. Raskhodnikova, et al. Strong Lower Bounds for Approximating Distribution Support Size and the Distinct Elements Problem. *SIAM Journal on Computing*, pages 813-842, 2009.

[3] G. Valiant, and P. Valiant. Estimating the Unseen: An n/log(n)-sample Estimator for Entropy and Support Size, Shown Optimal via New CLTs. In the *43rd ACM Symposium on Theory of Computing,* STOC, pages 685-694, 2011.

[4] S. Chaudhuri et al. Random sampling for histogram construction: How much is enough? In Proc. ACM SIGMOD International Conf. on Management of Data, pages 436–447, June 1998.

[5] P. J. Haas, et al. Sampling-based estimation of the number of distinct values of an attribute. In Proc. 21st International Conf. on Very Large Data Bases, pages 311–322, September 1995.

[6] G. Ozsoyoglu, et al. On estimating COUNT, SUM, and AVERAGE relational algebra queries. In Proc. Conf. on Database and Expert Systems Applications , pages 406–412, 1991.

[7] F. Olken. Random Sampling from Databases. PhD thesis, Computer Science, U.C. Berkeley, April 1993.

[8] P. B. Gibbons. Distinct-values estimation over data streams. In Manuscript, 2009.

[9] http://www.emcemearegistration.com/tapereplace/esquare/calculator.php - EMC Data Domain

[10] http://www.itcalc.com/ - NetApp Inc.

[11] https://www.snia.org/sites/default/files/Understanding_Data_Deduplication_Ratios-20080718.pdf

[12] P. Flajolet et al. Probabilistic counting algorithms for data base applications. J. Comput. Syst. Sci. 31(2): 182-209 (1985).

[13] N. Alon et al. The space complexity of approximating the frequency moments. ACM STOC, 1996, pp. 20–29.

[14] P. Flajolet, et al. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In AOFA 2007.

[15] D. M. Kane et al. An optimal algorithm for the distinct elements problem. In PODS, pp. 41–52, 2010.

[16] C. Estan, G. Varghese, and M. E. Fisk. Bitmap algorithms for counting active flows on high-speed links. IEEE/ACM Transactions on Networking, 14(5):925-937, 2006.

[17] P. B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. VLDB'01, pp 541–550

[18] Counting distinct items over update streams. ACM Journal Theoretical Computer Science 378(3):211-222, 2007

[19] D. Harnik et al. Estimation of deduplication ratios in large data sets. In Mass Storage Systems and Technologies (MSST), 2012

[20] IOzone Filesystem Benchmark http://www.IOzone.org/

[21] A. El-Shimi et al. Primary Data Deduplication -- Large Scale Study and System Design. USENIX ATC '12.

[22] D. Hitz et al. File system design for a file server appliance. In USENIX Technical Conference, 1994, pages 235–245

[23] P. Corbett el al. Row-Diagonal Parity for Double Disk Failure Correction. In Proceedings of the 2004 Usenix FAST, pp: 1- 14.

[24] E. Kave et al. A Framework for Analyzing and Improving Content-Based Chunking Algorithms No. HPL-2005-30R1.

[25] www.netapp.com/us/library/technical-reports/tr-3982.html

[26] A. Chen & A. Cao. Distinct counting with a self-learning bitmap. IEEE ICDE '09. Pages 1171–1174.

Appendix A

Denote the total number of distinct *fixed-size blocks* in the sample and data set as n_d and N_d , respectively.

Lemma 1. Assume the fingerprint has uniformity property and negligible collision probability. For large N_d , *err* has a normal distribution with zero as the mean and $(M-1)/N_d$ as the variance.

Proof: Because of good uniformity, any fingerprint in the data sets has $1/M$ probability to be sampled. Since the collision is negligible, the number of distinct fingerprints is approximately equal to the number of distinct blocks in the sample. The sampling can be treated as a Bernoulli trail of length N_d and successful rate $1/M$. The number of successes in the trail is equal to n_d. Therefore n_d follows a binomial distribution. Based on the definition of *err*, it can be represented as:

$$err = (R - R^*)/R = 1 - n_d \cdot M/N_d$$

When N_d is large, the distribution of n_d can be approximated as a normal distribution with $\mathrm{E}[n_d] = N_d / M$ and $\mathrm{Var}[n_d] = N_d \cdot (M-1)/M^2$. Therefore *err* follows a normal distribution of zero mean and $(M-1)/N_d$ as the variance. ■

Proof of Theorem 1: The sample in the variable-size blocks case can be seen as a group of K fixed-size block samples. According to Lemma 1, when N_i is large, n_i has a normal distribution: $\mathrm{E}[n_i] = N_i / M$ and $\mathrm{Var}[n_i] = N_i \cdot (M-1)/M^2$. Since S^* is a linear combination of n_i ($i = 1,\ldots,K$), S^* also follows a normal distribution with

$E[S^*] = \sum_{i=1}^{K} s_i \cdot N_i = S$ and $Var[S^*] = (M-1) \cdot \sum_{i=1}^{K} N_i \cdot {s_i}^2$.

Since $err = (S - \mathrm{S}^*)/\mathrm{S}$, it also has a normal distribution. E[*err*] is simply zero. The variance is calculated as:

$$Var[err] = \frac{Var[S^*]}{S^2} = (M-1) \cdot \sum_{i=1}^{K} N_i \cdot \frac{{s_i}^2}{S^2}$$

$$= \frac{(M-1)}{S} \cdot \sum_{i=1}^{K} s_i \cdot \frac{N_i \cdot s_i}{S} = \frac{(M-1) \cdot \bar{s}}{S}$$

This proves the theorem. ■

Appendix B

Proof: The proof of this theorem is simply based on the Empirical Rule of the normal distribution. Since the error has a normal distribution, the accuracy of the estimation has the following property.

$$Prob\left(|err| \geq \varepsilon \cdot \sqrt{Var[err]}\right) \leq 1 - erf\left(\varepsilon/\sqrt{2}\right)$$

, where *erf*() is the error function and ε is a constant. Substitute the variance of *err* from Theorem 1, we have:

$$Prob\left(|err| \geq \varepsilon \cdot \sqrt{(M-1) \cdot \bar{s}/S}\right) \leq 1 - erf\left(\varepsilon/\sqrt{2}\right)$$

We substitute ε with $\mathrm{erf}^{-1}(\beta) \cdot \sqrt{2}$ in the above inequality, which yields:

$$Prob\left(|err| \geq erf^{-1}(\beta) \cdot \sqrt{2} \cdot \sqrt{(M-1) \cdot \bar{s}/S}\right) \leq 1 - \beta$$

This means that as long as:

$$M \leq \frac{\alpha^2 \cdot S}{2 \cdot (erf^{-1}(\beta))^2 \cdot \bar{s}} + 1$$

the estimation satisfies α-β accuracy. ■