

Optimizing Fault-Tolerant System Using Post Priori Method

Jianbo Fan¹, Lidan Shou², Qingfeng Li¹, Jinxiang Dong²

¹College of Electron & Information Engineering, Ningbo University of Technology,
Ningbo Zhejiang 315016

²Institute of Artificial Intelligence, Zhejiang University, Hangzhou zhejiang 310027

Abstract

This paper presents a server performance model based on the approach of probabilistic analysis. This model is built on the experimental results using the X-Code in a network environment with stable overhead. The model effectively optimizes the mean service time for client data request by determining the optimal number of servers which store the original data when the total number of the servers are given and choosing the optimal read scheme, therefore, it improves the overall throughput and performance of distributed storage system. We note that this model is built without any a priori knowledge of the traffic information. The results indicate that in a distributed storage system, the method can play an active role, as proper data redundancy can help to improve not only the reliability of a storage system, but also the efficiency of it.

Keywords: Distributed Storage System, MDS Code, Performance, Redundancy.

1.0 Introduction

The array codes [1][2] have been under study by the researchers in communication and storage systems [3]. A common property of these codes is that the encoding and decoding procedures use only simple XOR and cyclic shift operations. In an array code, the information bits (original data) and parity bits (redundant data) are placed in an array of size $l \times n$. In a distributed storage system, the bits in a same column are stored in a same disk. If any bit in a disk fails, then this disk is deemed to be a failure disk and needs to repair. i.e. the code of

corresponding column is regard as an erasure. Current RAIN systems [4] can tolerate more than one disk failure at the same time.

In general, redundancy is a passive overhead to achieve reliability when redundancy becomes necessary for fault tolerance. However in this paper, the result of the experiments shows that data redundancy is an active part in the sense that redundancy can help to improve system performance (data throughput) of a distributed system.

Recently, Xiaodong Li et al [5] presented RDSS, a Resource Area Network (RAN)-based Distributed Storage System, which is designed for high scalability, long-term reliability, and operational efficiency. Using the RAN architectural pattern, RDSS works on the block level instead of the file level, which increases system flexibility. The contribution of RDSS is that it provides reliable storage services while achieving high efficiency. Marcos K. Aguilera et al [6] proposed a new approach to maintaining ensure-encoded data in a distributed system. The approach allows the use of space efficient k of- n erasure codes where n and k are large and the overhead $n-k$ is small. Concurrent updates and accesses to data are highly optimized.

We will adopt the technology of data redundancy based on MDS code to improve the performance of data server systems. Our server system is made up of a cluster of servers that are connected via a reliable communication network. In addition, broadcast is supported over the network, so that a client can broadcast its request for certain data to some or all of the n servers in the system. The data are distributed over the servers in such a way that a client can recover the complete requested data after it gets

*This work has been supported by the 2005 Scientific Research Plan Item of the education office of Zhejiang province in PR China under grant 20050012.

data from at least k of the n servers and this is true for any k servers. Such a distributed data server system is called a (n, k) server system and can be implemented by using the MDS code.

For a data server system, as the I/O bandwidth of local disks is much narrower than the CPU processing, the whole performance of the systems is dominated by the bottleneck of the disks. Distributed data over multiple servers can help to eliminate this bottleneck and improve the data throughput of the whole system, since data can be accessed in parallel from multiple servers at the same time. For a server system with n servers, in order to achieve best system performance, how should the optimal k_0 be determined when n is given? Once data redundancy is properly distributed among the servers, how should the optimal read scheme be chosen to optimize the mean service time?

Our main contribution is that in a network environment with stable overhead, we build a server performance model using the approach of probabilistic analysis to experiments and the experimental results based on X-Code. This model effectively optimizes the mean service time for client data request by determining the optimal number of servers which store the original data when the total number of the servers are given and choosing the optimal read scheme, increases the overall throughput and improves performance of distributed storage system.

2.0 The X code with MDS property

Reliability of distributed storage system is often achieved by storing redundant data in the systems using error-control codes. Generally, in a storage system, the failure of a single storage unit can be detected by the storage controllers and then can be masked as erasures. Erasure-correcting code is a mathematical method that can express data and recover lost data. An (n, k) erasure-correcting code can express k symbols of original data using n symbols of encoding data, namely $n-k$ is redundant quantity or parity check. What is called m -erasure-correcting code, i.e. original data can be recovered even though m symbols of encoding data are lost [7]. And this code of column distance d is defined as $d=m+1$. If $m=n-k$, then this code goes by the name of MDS (Maximum Distance Separable) code. MDS code is optimal according to redundant quantity

for the restoring ability of erasure.

Xu and Bruck [2] proposed a novel encoding algorithm, namely the X-code. In X-Code, information symbols are placed in an array of size $(n-2) \times n$. Parity symbols are constructed from the information symbols along several parity check diagonals with the XOR operation $+$. The parity symbols are placed in the last two rows of the array. So the array is of size $n \times n$ and each column has information symbols as well as parity symbols. Let $C_{i,j}$ be the symbol at row i and column j . The parity symbols of X code are defined as:

$$C_{n-2, i} = \sum_{k=0}^{n-3} C_{k, \langle i+k+2 \rangle_n}, C_{n-1, i} = \sum_{k=0}^{n-3} C_{k, \langle i-k-2 \rangle_n} \quad (1)$$

Where $i=0,1,\dots,n-1$, and $\langle x \rangle_n = x \bmod n$.

From formula (1), it is easy to see that each parity symbol at row $n-2$ and $n-1$ is just the checksum of the information symbols along diagonals of slopes -1 and 1 respectively. The theorem 1 of X-code with MDS property is described below.

Theorem 1: X-code has column distance of 3, i.e., it is MDS, if and only if n is a prime number.

Details about X-code and the proof of the theorem 1 is omitted in this paper. They can be found in [2].

3.0 Server performance analysis

Before presenting our method, we need to define a server system model based on the technique of probabilistic analysis.

3.1 Server system model

Definition1: Define the service time T_i of the server i ($1 \leq i \leq n$) to be the elapsed time from when the client sends its request to the server i to when it receives data from the server i .

Notice that T_i does not include the time needed at the client side to do any necessary computations to recover the final data, since we assume that the computations are rather simple and take much less time than does the data delivery through communication media. We model T_i as a continuous random variable with probability density function (*pdf*) $f_i(t)$. For simplicity of analysis, we assume that all T_i s are independent and identically distributed random variable, i.e. $f_i(t)=f(t)$, $1 \leq i \leq n$.

3.2 Analysis result

Let $F_i(t)$ be the cumulative distribution function (*cdf*) of T_i :

$$F_i(t) = \text{Probability}(T_i \leq t) = \int_0^t f_i(x) dx \quad (2)$$

Definition 2: Let $T(n, k)$ be the elapsed time from when the client broadcasts its data request to the servers to when it receives data from at least k out of the n servers.

Thus $T(n, k)$ is another random variable and is a simple function of all the T_i s: $T(n, k) \geq T_i$, $n \geq k$, $1 \leq i \leq n$. Let $f_{(n,k)}(t)$ and $F_{(n,k)}(t)$ be the *pdf* and *cdf* of $T(n, k)$ respectively, then it is easy to relate $F_{(n,k)}(t)$ and $f_{(n,k)}(t)$ with $F(t)$ and $f(t)$ [8]:

$$F_{(n,k)}(t) = \sum_{i=k}^n \binom{n}{i} F(t)^i [1-F(t)]^{n-i} \quad (3)$$

And $f_{(n,k)}(t)$ can be gotten by calculating:

$$f_{(n,k)}(t) = \frac{dF_{(n,k)}(t)}{dt} = k \binom{n}{k} F(t)^{k-1} [1-F(t)]^{n-k} f(t) \quad (4)$$

The mean of $T(n, k)$, $E[T(n, k)]$, is a good measurement of the server system's performance. It can be calculated once the $f_{(n,k)}(t)$ is known:

$$E[T(n, k)] = \int_0^{\infty} t f_{(n,k)}(t) dt \quad (5)$$

3.3 Properties of mean service time

For a fixed *pdf* $f(t)$ and *cdf* $F(t)$, the expression $F(t)^i [1-F(t)]^{n-i}$ in formula (3) is a limitary quantity because of $0 < F(t) < 1$. Intuitively, a bigger n and /or a smaller k leads to a bigger $F_{(n,k)}(t)$. In addition, a bigger n and /or a smaller k leads to a smaller $E[T(n, k)]$ and this can be declared as a theorem 2.

Theorem 2: For a continuous random variable $T[n, k]$ be defined on $[a, b]$ with a fixed *pdf* $f_{(n,k)}(t)$ and *cdf* $F_{(n,k)}(t)$, the following inequalities hold for $1 \leq k \leq n$:

1. $E[T(n, k)] > E[T(n+m, k)]$, for $m \geq 1$;
2. $E[T(n, k)] < E[T(n, k+m)]$, for $m \geq 1$;
3. $E[T(n+m, k)] < E[T(n, k+m)]$, for $m \geq 1$;

The above theorem 2 can be easily proved by using formula (5) and (4) and adopting the characteristic of formula (3) and the fact that $F_{(n,k)}(a) = F_{(n,k)}(b) = 0$. About proof of the theorem 2 may read reference [8]. Here is omitted. We will use these properties as guidelines to determine the optimal k_0 and choose the optimal read scheme for optimizing the mean service time.

From formula (4) and (5), $E[T(n, k)]$ is a function of the *pdf* $f(t)$ of an individual server's data service time. Our goal is to reduce $E[T(n, k)]$ under various conditions. The data service time T depends on many factors in a practical server system, such as CPU speed of the server and the client, local disk I/O speed of the servers and bandwidth and latency of the communication medium connecting the server and client. Here we will try to model the data service time as a simple probability distribution, it can be analyzed rather easily, and yet can approximate the real data service time closely. Such a model will be abstracted from the experimental results of a real data server system.

4.0 The *post priori* method

In this section, we firstly present the experimental results collected from a prototype system, which implements the proposed techniques. We shall then give the optimized server model based on the experimental data.

4.1 Experimental Settings

Our experimental server system consists of several x86 servers running Windows XP. Each server has data stored on its local disk. The client is also a PC running the same OS. The nodes are connected via CableTron Systems Smart Switch Router 8600. Experiments are conducted in such a real system to measure the service time for data of different size. The procedure of the experiment is as following:

- (1) The client sends a request for a certain amount of data to a server;
- (2) The server reads the data from its local disk and sends it to the client through the reliable communication layer;
- (3) The data are delivered to the client through the reliable communication layer.

The data service time is measured from the instant that the client finishes sending its request to the instant that the client gets the data.

The figure 1 below shows a (n, k) server system that is described in section 1.

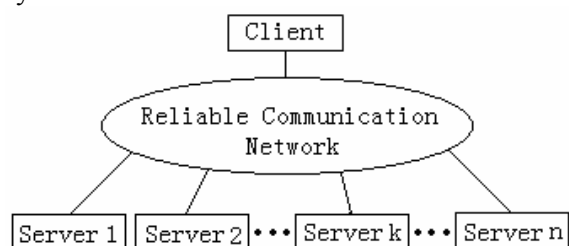


Figure 1 a (n, k) server system

In a stable network environment with little traffic, we run programs of the data service time a few thousand times for data of a given size, and get some original data and fitting curves according to the observed frequencies of different ranges of service time. Table 1 shows data of the service time and empirical *pdfs* for data of different size. At the column of Service Time in Table 1, the value in the front or rear of symbol \pm shows the value of the centroid of the data cluster and the maximum offset in terms of weighted average respectively. At the column of Empirical *pdfs* in Table 1, the value indicates the times of data service time fallen in interval $[c, d]$. With the help of software Origin 7.0, we use wave form function to fit these data of Table 1 and get three curves shown in Figure 2 below.

Table 1 data of the service time and empirical *pdfs* for data of different size

Data size	Service Time	Empirical pdfs
32K bytes	0.00313 \pm 0.00013	90 in [0.0030,0.0032]
	0.00627 \pm 0.00027	450 in [0.0060,0.0064]
	0.00937 \pm 0.00023	400 in [0.0092,0.0096]
	0.01247 \pm 0.00013	60 in [0.0124,0.0126]
320K bytes	0.05935 \pm 0.00025	125 in [0.0592,0.0596]
	0.06246 \pm 0.00034	380 in [0.0622,0.0628]
	0.06563 \pm 0.00023	485 in [0.0654,0.0658]
3200K bytes	0.0718 \pm 0.001	10 in [0.0708,0.0728]
	0.38753 \pm 0.00027	75 in [0.3874,0.3878]
	0.39063 \pm 0.00023	425 in [0.3904,0.3908]
	0.39376 \pm 0.00016	485 in [0.3936,0.3938]
	0.39793 \pm 0.00207	15 in [0.3968,0.4000]

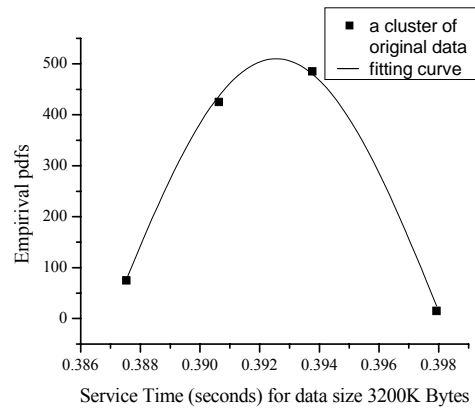
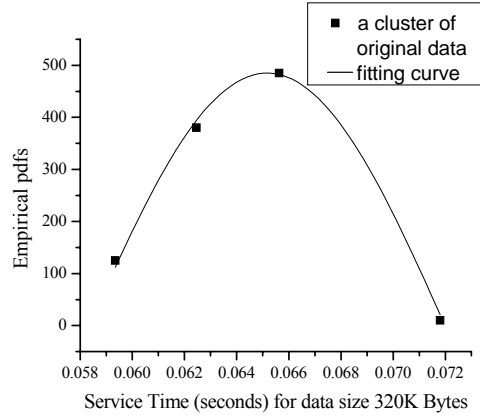


Figure 2 Curves of the service time and empirical *pdfs* for data of different size

The experimental results show that the shapes of service time and empirical *pdfs* for data of different size can be approximated by the same probability density function like a wave form distribution, the size of distributed base is proportional to the data size basically, and the width of the distributed base is almost the same for data of different size. The result is obtained as follows: the data service time T is regarded as a random variable defined on $[a, b]$ (a and b are two parameters of a real system), the *pdf* of the data service time T is fit as :

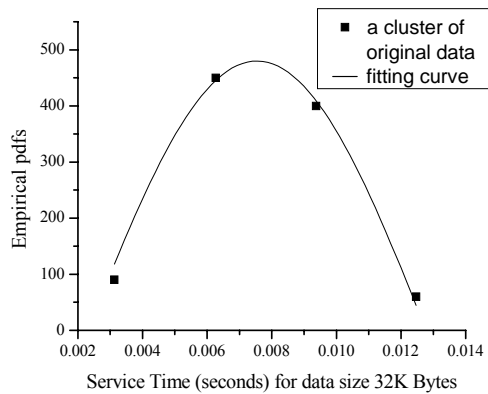
$$f(t) = \frac{\pi}{2(b-a)} \sin\left(\frac{t-a}{b-a}\pi\right), \quad t \in [a, b] \quad (6)$$

Obviously, $f(t)$ satisfies the two characters of *pdf* and the *cdf* of T may be calculated as:

$$F(t) = \frac{1}{2} - \frac{1}{2} \cos\left(\frac{t-a}{b-a}\pi\right), \quad t \in [a, b] \quad (7)$$

4.2 Analysis of mean service time

In a server system with n servers, we need to determine the number k of the servers which



store the original data in order to maximize the performance of the whole system. i.e. to minimize the mean service time of client's data request. If k is given, then the rest of the servers can store the redundancy data. When n and the pdf $f(t)$ are fixed, $E[T(n, k)]$ decreases monotonically as k decreases according to theorem 2. This means that in order to make $E[T(n, k)]$ small, k should be as small as possible. On the other hand, the smaller k is, the more data needs to be stored on each server, since the total amount of the data that a client needs is always fixed. This means higher service time from each server. Our goal is to find such an optimal k_0 that when both side of the problem are considered, $E[T(n, k)]$ is minimized.

Applying the pdf model of each server's service time T and using MDS codes for distributing the redundant data, we obtain that if the pdf of T is $f(t)$, $t \in [a, b]$ when $k=1$, then for general k , the corresponding pdf is $f(t)$, $t \in [a/k, b/k]$, since the size of distributed base of the pdf is proportional to the data size basically. In practice, the system parameters a and b can be determined by experiments, thus $E[n, k]$ can be calculated according to formula (5). When $a=1$ and $b=2$, we get optimal k_0 by calculating $E[n, k]$ in the case of ideal ($1 \leq k \leq n$). For example, when $n=5, 10, 15, 20, 25, 30$; $k_0 = 5, 10, 15, 18, 18, 18$ correspondingly. Even above examples use specific pdfs, the same method is also applicable with other pdfs by defining suitable $f(t)$ into formula (5). For a given server system, such an optimal k_0 can always be found theoretically. Proper MDS codes can then be used based on the (n, k) pair. Thus we obtain an optimal data distribution scheme for a given server system.

4.3 Data acquisition scheme based on X-code

Once the data distribution scheme is set, i.e. k is determined and the proper MDS array code is chosen, the client needs to decide how to request data. In general, a client should send its request to as many servers as possible and also make the amount of data it needs from each server as small as possible, since the properties of $E[T(n, k)]$ show that more redundancy brings better performance theoretically. For a specific distribution scheme, the client needs to calculate the mean service time of all possible data read scheme, and then choose an optimal read scheme. Since the read scheme is closely related

to the MDS array code being used, here we will give an example using X-code to show the guidelines for choosing an optimal read scheme.

In this example, the server system has n servers, and the data that the client requests can be assembled from any $n-2$ servers, i.e. this is a $(n, n-2)$ system. The X-code can be used to implement this system. The data distribution using X-code is as follows: (1) the whole original (information) data is partitioned into $(n-2)*n$ blocks of equal size; (2) each of the n servers stores $n-2$ blocks of original data and 2 blocks of parity data; (3) $2*n$ blocks of redundancy (or parity) data are calculated according to the encoding rules of the X-code.

The MDS property of the X-code may have 3 read schemes for reconstructing the whole original data from the data stored on n servers: (a) Read from all of n requested servers, each of which sends its $n-2$ blocks of original data. Let $E[T(n, n)]_{n-2}$ denoted the mean data service time of the Scheme 1. (b) Read from any $n-2$ requested servers, each of which sends its $n-2$ blocks of original data and 2 blocks of parity data. Let $E[T(n-2, n-2)]_n$ denoted the mean data service time of the Scheme 2. (c) Read from any $n-2$ of n requested servers, each of which sends its $n-2$ blocks of original data and 2 blocks of parity data. Let $E[T(n, n-2)]_n$ denoted the mean data service time of the Scheme 3.

Notice that there is no redundant data in scheme 1 or scheme 2, so the client must wait until it receives all the data from all the servers. But in scheme 3, there is redundant data, thus the client only needs to receive data from any $n-2$ of the n requested servers. From property 1 of Theorem 2, $E[T(n-2, n-2)]_n > E[T(n, n-2)]_n$. But the relation between $E[T(n, n)]_{n-2}$ and either $E[T(n-2, n-2)]_n$ or $E[T(n, n-2)]_n$ is not so obvious, since in scheme 1 the client needs to wait for more servers, but needs less data (i.e. less service time) from each server. To determine which scheme is best scheme for a given system, we need calculate the mean of the whole service time for all possible schemes.

Assume that the pdf of the service time T for each server to send n blocks of data to the client is $f(t)$, $t \in [a, b]$; then the pdf of T in scheme(1) is $f(t)$, $t \in [a(n-2)/n, b(n-2)/n]$, since each server only needs to send $n-2$ blocks of data, and the pdf of T in scheme (2) or (3) is $f(t)$, $t \in [a, b]$. Now the mean of the entire service time in the different schemes can be calculated according to formula (4), (5), (6) and (7). As an example,

assume $a=1$ and $b=2$, then Table 2 shows the mean service time of three schemes for different n .

Table 2 The mean service time of three schemes

N	Scheme 1 $E[T(n, n)]_{n-2}$	Scheme 2 $E[T(n-2, n-2)]_n$	Scheme 3 $E[T(n, n-2)]_n$
10	1.458384	1.802961	1.657425
20	1.685685	1.866458	1.760814
30	1.768622	1.891549	1.805458
40	1.812390	1.905649	1.831837
50	1.839642	1.914826	1.849756
60	1.858294	1.921301	1.862946
70	1.871865	1.926105	1.873178
75	1.877347	1.928065	1.877504
76	1.878360	1.928428	1.878317
80	1.882167	1.929793	1.881415
90	1.890233	1.932689	1.888230
100	1.896698	1.935001	1.893991
110	1.901974	1.936867	1.898944
120	1.906341	1.938384	1.903262

From above calculation, the performance of the three read schemes depend on the parameter n when a and b are fixed. When $n \leq 75$, mean service time of the scheme 1 is optimal among the three schemes. But when $n \geq 76$, mean service time of the scheme 3 is optimal among the three schemes.

The above example shows that the client has different ways of reading data from the servers after the data distribution is set at the server side.

5.0 Conclusion and future work

In this paper, we proposed a server performance model. We have conducted extensive experiments in this server model, and obtained the following results: (1) The optimal k_0 gotten by calculating $E[T(n, k)]$ is a tradeoff when n and $f(t)$ are given; (2) Our experimental results based on X-Code shows that the $E[T(n, n-2)]_n$ of the Scheme 3 is optimal as n increases ($n \geq 76$), and $n-k=n-(n-2)=2$. The result is different from the work in [6] in that: [6] proposed a new approach to maintaining ensure-encoded data in a distributed system and our approach allows the use of space efficient k of- n erasure codes where n and k are large and the overhead $n-k$ is small.

Future work includes that finding a new MDS code which may correct more than two erasures. For a given system (i.e. a certain the pdf of T , a fixed (n, k) pair and a particular code), there always exists an optimal read scheme for the client. Since data read schemes are highly related to the codes used, exploring new MDS code with optimal performance is an interesting and challenging research problem.

References

- [1] M.Blaum, J.Bruck, A.Vardy, "MDS Array Codes with Independent Parity Symbols", IEEE Trans. On Information Theory, 42(2), p529-542, March 1996.
- [2] L.Xu, and J.Bruck, "X-Code: MDS Array Codes with Optimal Encoding", IEEE Transactions on Information Theory, 45(1), p272-276, Jan. 1999.
- [3] P.G. Farrell, "A Survey of Array Error Control Codes", Europ. Trans. Telecommun., 3(5), p441-454, 1992.
- [4] V. Bohossian, C.Fan, P.LeMahieu, M.Riedel, L.Xu and J.Bruck, "Computing in the RAIN: A Reliable Array of Independent Notes", IEEE Trans. On Parallel and Distributed Systems, 12(2), p99-114, Feb. 2001.
- [5] Xiaodong Li and Chang Liu ; Towards a Reliable and Efficient Distributed Storage System ; System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on Jan. 2005 Page(s): 301c - 301c.
- [6] Marcos K. Aguilera et al, "Using Erasure Codes Efficiently for Storage in a Distributed System", Proceedings of International Conference on Dependable Systems and Networks, DSN 2005, p336 - 345, 28 June-1 July 2005.
- [7] F.J.MacWilliams and N.J.A.Sloane, "The Theory of Error Correcting Codes", Amsterdam: North-Holland, 1977.
- [8] L.Xu, and J.Bruck, "Highly Available Distributed Storage Systems", Proceedings of Workshop on Distributed High Performance Computing, Lecture Notes in Control and Information Sciences, Springer Verlag, 1999