Chapter 5 Improving the Performance of Data Servers

5.1 Introduction

It has become common to use a cluster of distributed computing nodes in server systems, such as image servers, video servers, multimedia servers, and web servers, all of which can be regarded as specific kinds of the more general concept of data servers. Distributed server systems can improve both data *reliability* (or availability) and *performance* (or efficiency, i.e., data throughput) of the system. Much research, such as the well known k-out-of-n systems[3], has been done to improve reliability by introducing *data redundancy* or *information dispersity*[27] into systems.

In a real system, although the redundant data enables the system to provide continuous service when certain failures (communication link failures, server node failures) occur, most of the time the system works in a normal mode, i.e., there is no failure in the system; in this case data redundancy can be used to improve the performance of the system. It was first shown in [11] that redundant data can make packet routing more efficient by reducing the mean and variance of the routing delay. Recently, more scalable and efficient reliable multicast schemes have also been proposed based on data redundancy in the messages to be multicast[14].

In this chapter, we propose a method based on error-correcting codes for improving the performance of services in general data server systems. Our system setup is shown in Figure 5.1: a cluster of servers is connected via some 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 is distributed over the servers in such a way that a client can recover the complete requested data after it gets 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 an (n, k) server system in this chapter. Again, such (n, k) systems can be implemented by using error-correcting codes, particularly MDS array codes.

For a data server system, since the I/O speed of the local disks is much slower than

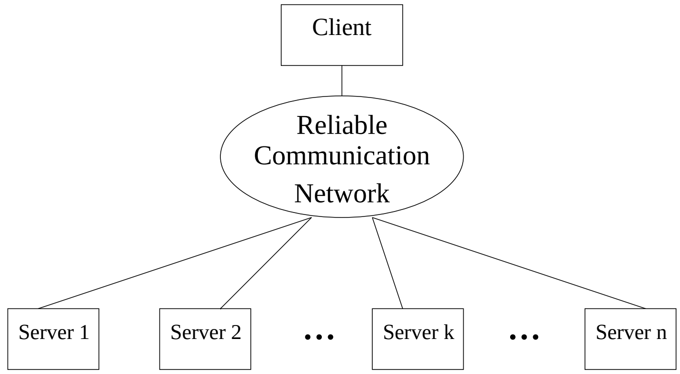


Figure 5.1: An (n, k) server system

the CPU speed of the servers, the whole performance of the system is dominated by the bottleneck of the disks. Distributing data over multiple servers can help overcome this bottleneck and improve the data throughput of the whole system, since the data can be accessed in parallel from multiple servers at the same time. For a server system with n servers, the system performance can be further improved by introducing proper data redundancy into the system, i.e., an (n, k) system should be properly chosen instead of naively distributing the raw data over all the n servers. This was shown in Example 1.5 and Example 1.6 in Chapter 1. What is the *proper* redundancy when the total number of the servers is given? Or how should k be determined when n is given, in order to achieve the best system performance? This is the so-called *data distribution* problem at the server side, which will be discussed in detail later in this chapter. Also, as already discussed in Example 1.7 in Chapter 1, there is another problem called the *data acquisition* at the client side: that once data redundancy is properly distributed among the servers, how should matching read approaches be chosen to optimize the mean service time? This problem will be also explored in this chapter. The main contribution of this chapter is to propose data distribution and acquisition schemes for a given server system that improve the system performance.

This chapter is organized as follows. Section 5.2 first describes a probability model of distributed server systems, then gives analytical results about the service time of a general distributed data server system. In Section 5.3, experimental results are used to create a probability model for service time in a practical disk-based data server system. The data distribution and data acquisition schemes are discussed in more detail in Section 5.4. Section 5.5 concludes the chapter and proposes a future research direction.

5.2 Preliminary Analysis

Before we consider other problems, we first define a server system model we will be using. Then we give some basic analytical results that can be used further to solve the data distribution and data acquisition problems.

5.2.1 System Model

In this chapter, a distributed server system consists of n servers. A client can broadcast its request for certain data to all the servers. All communications among the client and servers are reliable, i.e., there is no packet loss, order change or content corruption. Each server stores a portion of the requested data in such a way that the client can recover its requested data after it receives data from at least k ($k \leq n$) out of the n servers. 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 here we assume that the computations are rather simple and thus 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) $f_i(t)$[23]. For simplicity of analysis, we assume that all T_is are i.i.d (*independent, identically distributed*) random variables, i.e., $f_i(t) = f(t)$, $1 \leq i \leq n$.

5.2.2 Analysis Results

Let $F_i(t)$ be the *cumulative distribution function* (cdf) of T_i, i.e.[23],

$$F_i(t) = Probability(T_i \leq t) = \int_0^t f_i(x)dx$$

Now 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. Then $T(n, k)$ is another random variable and is a simple function of all the T_is:

$$T(n, k) \geq T_i, \quad where \ \|\{i\}\| \geq k$$

In the above equation, $\|S\|$ is the number of the elements in the set S.

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 $f_{(n,k)}(t)$ to $F(t)$ and $f(t)$ [11]:

$$F_{(n,k)}(t) = \sum_{i=k}^{n} \binom{n}{i} F(t)^i [1 - F(t)]^{n-i} \tag{5.1}$$

or [11][32]:

$$f_{(n,k)}(t) = \frac{dF_{(n,k)}(t)}{dt} = k \binom{n}{k} F(t)^{k-1} [1 - F(t)]^{n-k} f(t) \tag{5.2}$$

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:

$$E[T(n,k)] = \int_0^\infty t f_{(n,k)}(t) dt \tag{5.3}$$

5.2.3 Properties of Mean Service Time

Though it is usually hard to get a clean closed form of $E[T(n,k)]$ for a general pdf $f(t)$, it is still possible to get some of its properties with respect to n and k. Intuitively, for a fixed pdf $f(t)$, a bigger n and/or a smaller k leads to a smaller $E[T(n,k)]$ and this can be proven mathematically.

Before we discuss properties of the $E[T(n,k)]$, we give a lemma which can be used to prove the properties.

Lemma 5.1 *Let two continuous random variables X and Y be defined on $[a,b]$ with cdf's $F_X(t)$ and $F_Y(t)$ respectively. If $F_X(t) \geq F_Y(t)$, for all t, $a \leq t \leq b$, and $F_X \not\equiv F_Y$, i.e., the pdf of X is* left *of that of Y, then $E[X] < E[Y]$.* $\square$

Proof: Notice that $F_X(b) = F_Y(b) = 1$ and $F_X(a) = F_Y(a) = 0$, then

$E[X] - E[Y] = \int_a^b t dF_X(t) - \int_a^b t dF_Y(t) = [tF_X(t)|_a^b - \int_a^b F_X(t)dt] - [tF_Y(t)|_a^b - \int_a^b F_Y(t)dt]$
$= t[F_X(t) - F_Y(t)]|_a^b - \int_a^b [F_X(t) - F_Y(t)]dt = -\int_a^b (F_X(t) - F_Y(t))dt < 0.$ $\square$

It has been shown in [27] that

Lemma 5.2 *For a random variable T with a fixed pdf $f(t)$, the following inequalities hold for $1 \leq k \leq n$ and for $0 < F(t) < 1$:*

1. *$F_{(n,k)}(t) < F_{(n+m,k)}(t)$, for $m \geq 1$;*

2. $F_{(n,k)}(t) > F_{(n,k+m)}(t)$, *for* $m \geq 1$;

3. $F_{(n,k)}(t) > F_{(n+m,k+m)}(t)$, *for* $m \geq 1$;

4. $F_{(i,j)}(t) \leq F_{(n,k)}(t)$, *if* $n \geq i$ *and* $k \leq j$, *equality holds only when* $n = i$ *and* $k = j$;

5. $F_{(i,j)}(t) > F_{(n,k)}(t)$, *if* $n \geq i$, $k > j$ *and* $n - k \leq i - j$.

□

Using the two lemmas, it is straight forward to get the following properties of the mean of the service time:

Theorem 5.1 *For a random variable* T *with a fixed pdf* $f(t)$, *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. $E[T(n,k)] < E[T(n+m,k+m)]$, *for* $m \geq 1$;

4. $E[T(i,j)] \geq E[T(n,k)]$, *if* $n \geq i$ *and* $k \leq j$, *equality holds only when* $n = i$ *and* $k = j$;

5. $E[T(i,j)] < E[T(n,k)]$, *if* $n \geq i$, $k > j$ *and* $n - k \leq i - j$.

□

We will use these properties as guidelines in Section 5.4 for the data distribution. One would hope that the *variances* of random variables also had the similar properties. Unfortunately, however, the above properties do *not* hold for the variances. We will show one example about the variances and one more property of the $E[T(n,k)]$.

Lemma 5.3 *Let two continuous random variables* X *and* Y *be defined on* $[a,b]$ *with pdf's* $f(t)$ *and* $g(t)$, *respectively. If* $f(t) = g(a+b-t)$, *for all t,* $a \leq t \leq b$, *i.e.,* $f(t)$ *is the* reflection *of* $g(t)$ *about the line* $t = \frac{a+b}{2}$, *then* $E(X) = a + b - E(Y)$, *and* $Var(X) = Var(Y)$, *where* $Var(X)$ *and* $Var(Y)$ *are the* variances *of* X *and* Y. □

Proof: Straight forward, omitted.

Lemma 5.4 *If the pdf $f(t)$ of a random variable T defined on $[a, b]$ is* symmetric *or* self-reflective *about the line $t = \frac{a+b}{2}$, i.e., $f(t) = f(a+b-t)$, then $f_{(n,k)}(t) = f_{(n,n+1-k)}(a+b-t)$.* □

Proof: First, it is easy to show that if $f(t) = f(a + b - t)$, then

$$F(t) = 1 - F(a + b - t) \tag{5.4}$$

Using Eq.(5.2), Eq.(5.4) and the identity $k\binom{n}{k} = (n + 1 - k)\binom{n}{n+1-k}$, we can get

$$f_{(n,n+1-k)}(a + b - t) = (n + 1 - k)\binom{n}{n+1-k} F(a + b - t)^{n-k}[1 - F(a + b - t)]^{k-1} f(a + b - t)$$

i.e.,

$$f_{(n,n+1-k)}(a + b - t) = k\binom{n}{k}[1 - F(t)]^{n-k} F(t)^{k-1} f(t) = f_{(n,k)}(t)$$

□

This lemma shows that if T's pdf is symmetric, then there is also symmetry between $T(n, k)$ and $T(n, n+1-k)$: their pdfs are reflections of each other. The above two lemmas lead to following theorem:

Theorem 5.2 *If the pdf $f(t)$ of a random variable T, defined on $[a, b]$ is* symmetric *about the line $t = \frac{a+b}{2}$, i.e., $f(t) = f(a + b - t)$, then $E[T(n, n + 1 - k)] = a + b - E[T(n, k)]$, and $Var[T(n, k)] = Var[T(n, n + 1 - k)]$.* □

Here we see an example where the monotonicity of $E[T(n, k)]$ with respect to n or k does *not* hold for $Var[T(n, k)]$.

5.3 Server Performance Model

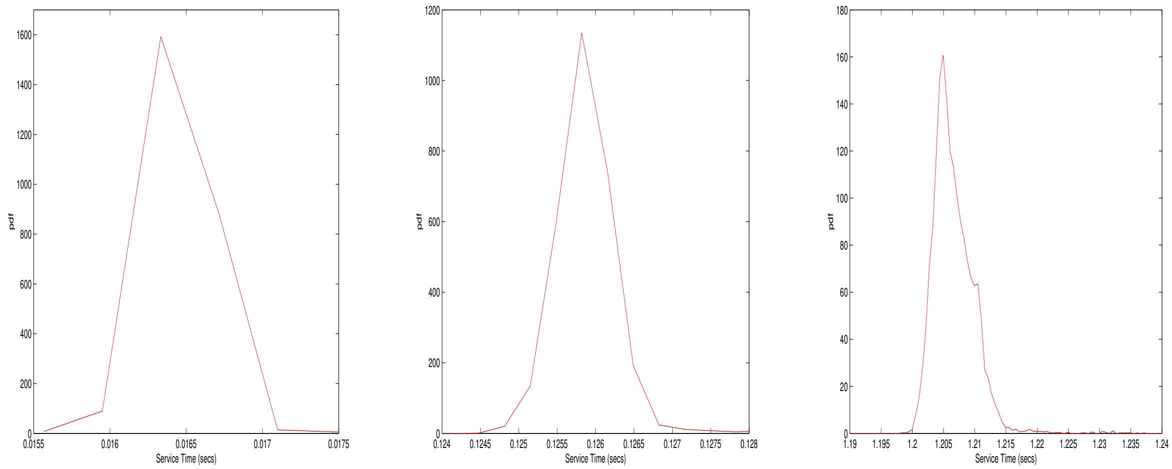
From Eq.(5.2) and Eq.(5.3), $E[T(n, k)]$ is a function of the pdf $f(t)$ of an individual server's data service time. The goal of the data distribution and data acquisition problem is to reduce $E[T(n, k)]$ under various conditions. Before we analyze the data distribution and data acquisition problem, it is necessary to establish some model of $f(t)$.

5.3.1 Abstraction from Experiments

The data service time T depends on many factors in a practical server system, such as computing power (i.e., CPU speed) of the servers and the client, local disk I/O speed of the servers and bandwidth and latency of the communication medium (usually including a reliable communication software layer) connecting the servers and the client. A model considering all the factors will be fairly complex. In this chapter, we will try to model the data service time as a simple probability distribution, that can be analyzed rather easily, and yet can approximate the real data service time closely. Such a model will be abstracted from experimental results of a real data server system.

Our experimental server system consists of several servers, which are PCs running Linux. Each server has data stored on its local hard disk. Data is accessed via the Linux file system. The client is also a PC running the same Linux. The nodes are connected via Myrinet switches. A *sliding window* protocol is used to ensure reliable communication. Experiments are conducted in such a real system to measure the service time for data of different sizes. The procedure of the experiment is as follows: (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 is 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. We run the above procedure a few thousand times for data of a given size, and get the service time pdf according to the observed frequencies of different ranges of service time. Figure 5.2 shows empirical service time pdfs for data sizes (a) 32 Kbytes, (b) 320 Kbytes and (c) 3200 Kbytes.

The effective data bandwidths in this experiments are quite low, since they are the concatenation of the local disk bandwidth and the reliable communication layer bandwidth. But the shapes of the bandwidth pdfs are more interesting. The experiment results show that the shapes of empirical pdfs of different data size can be approximated by the same distribution. A closer look shows that the width of the distribution base is approximately *proportional to* the data size. More complex distributions, such as the Gamma distribution or the Beta distribution, might give more accuracy. But to simplify the analysis, that follows, we will regard the data service time T as a random variable defined on $[a, b]$ (a



(a) $data\ size = 32Kbytes$ (b) $data\ size = 320Kbytes$ (c) $data\ size = 3200Kbytes$

Figure 5.2: Empirical pdfs of service time for data of different sizes

and b are two parameters of a real system), which follows a triangular distribution, denoted $Tr[a, b]$:

$$f(t) = \begin{cases} \frac{4(t-a)}{(b-a)^2} & a \leq t \leq \frac{a+b}{2} \\ \frac{4(b-t)}{(b-a)^2} & \frac{a+b}{2} < t \leq b \end{cases} \tag{5.5}$$

Its cdf (*cumulative distribution function*) is

$$F(t) = \begin{cases} \frac{2(t-a)^2}{(b-a)^2} & a \leq t \leq \frac{a+b}{2} \\ 1 - \frac{2(b-t)^2}{(b-a)^2} & \frac{a+b}{2} < t \leq b \end{cases} \tag{5.6}$$

One explanation for this model is as follows: in a real system, data is delivered in packets of some small size. The delivery time of the ith packet is a random variable t_i, whose probability distribution can be characterized by a uniform distribution over some time span; the t_i's are assumed to be i.i.d. random variables. Then the service time T of the whole data is: $T = s + \sum_i t_i$, where s is another uniform random variable describing the setup (or overhead) time for sending a certain amount of data. Thus the pdf of T is a *Gaussian-like* function, whose base width is approximately proportional to the number of the packets in the data, which in turn is proportional to the data size. For simplicity, we approximate the Gaussian-like function by a suitable triangular function. The distributions are shown in Figure 5.3.

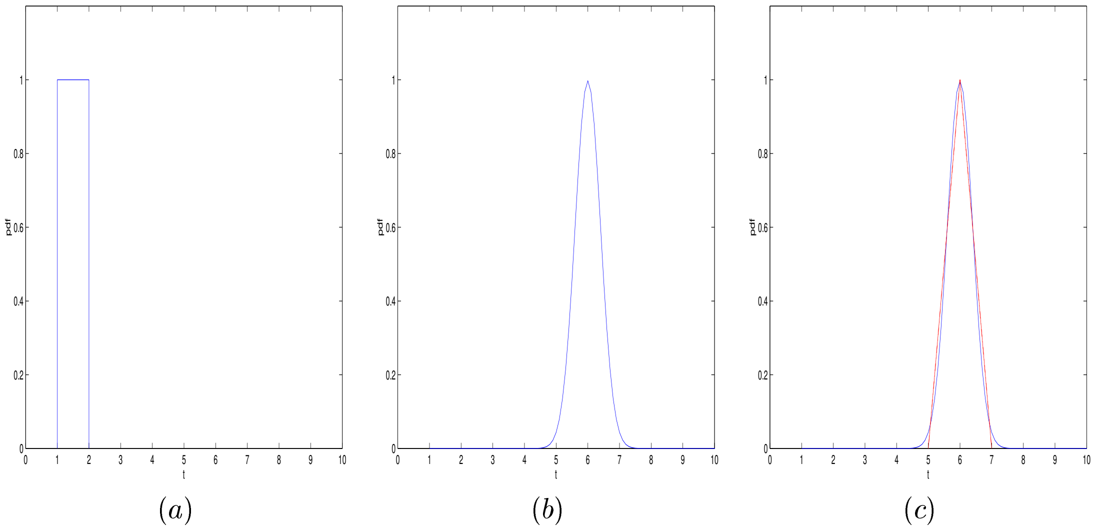
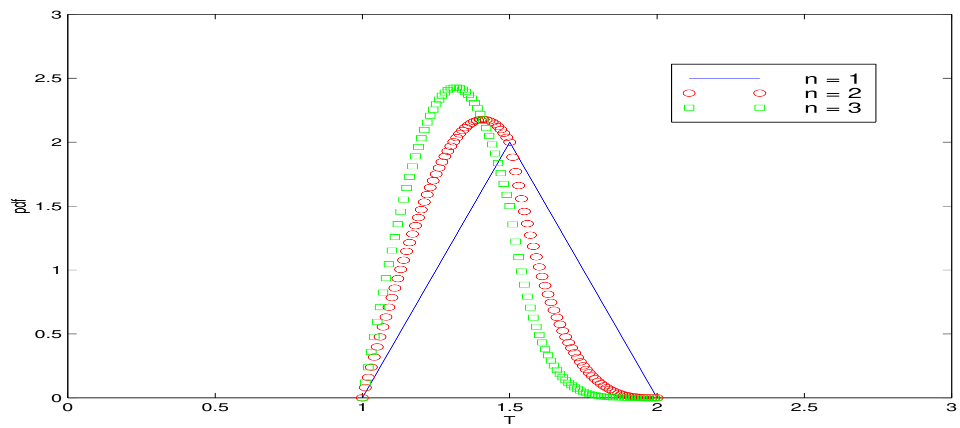


Figure 5.3: Probability distributions of data service time of (a) single packet, (b) the whole data, (c) the approximation with $Tr[a,b]$

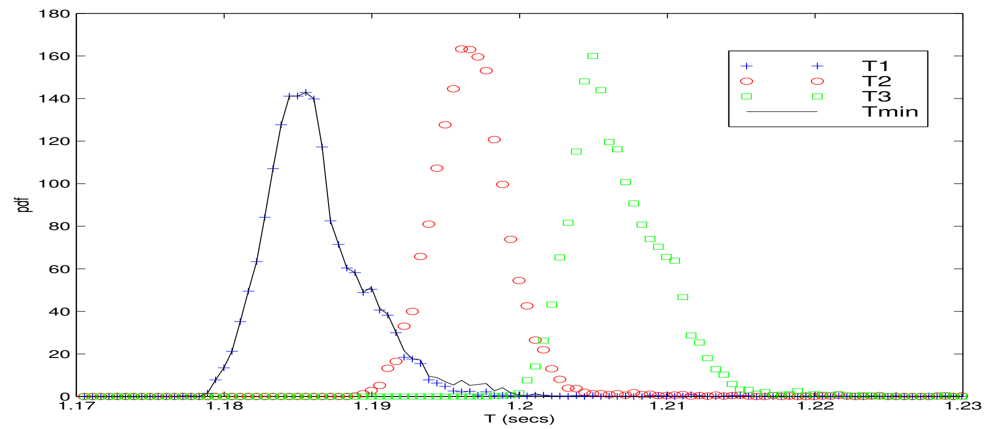
5.3.2 Verification with T(n,1)

Intuitively, having more servers should provide better performance when the amount of data stored on each server is fixed, i.e., $E[T(n,k)]$ decreases as n increases and/or k decreases. We can get pdfs of the $T(n,k)$ for a data server system by evaluating Eq.(5.2) for the service time distribution in Eq.(5.5) and Eq.(5.6). Figure 5.4(a) shows the pdfs of $T(n,1)$, where $1 \leq n \leq 3$ and T is of the triangular distribution $Tr[1,2]$. Here we can see the pdf of $T(n,k)$ shifts left as n increases.

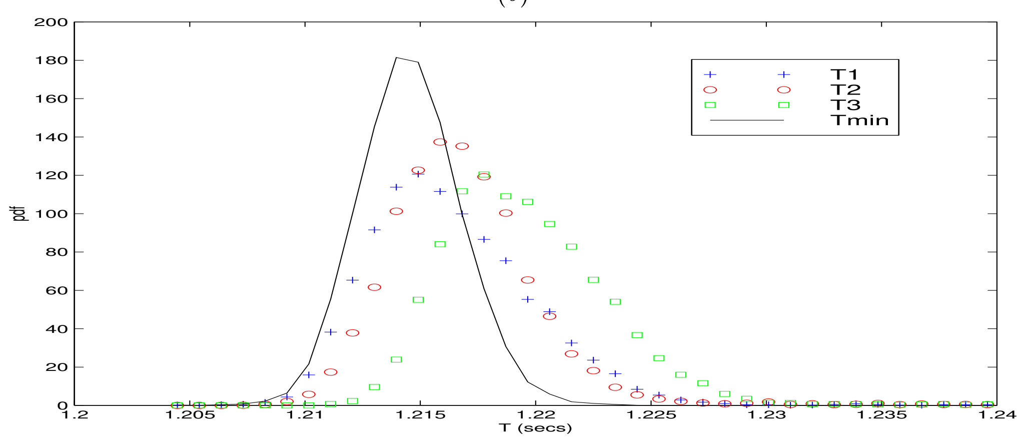
To further verify the properties of $E[T(n,k)]$, simple experiments to measure $T(n,1)$ were done on the experimental server system described in previous subsection. The system consists of three servers. In order to remove other factors that also affect data service time, such as contention in the communication medium (including the reliable communication layer, which is a bottleneck if we use a single client which communicates with the three servers), we use three clients, each of which is served by a separate server. Conceptually the three clients are regarded as a single client, thus the whole data service time is the minimum of the three individual service time of the server-client pairs. Figure 5.4(b) shows the service times (T_1, T_2, and T_3) of the three individual server-client pairs for 3200 Kbytes data each. Since the variance among the three pairs is bigger than the variance within each pair, the whole service time (T_{min}), which is the minimum of the three, is determined by



(a)



(b)



(c)

Figure 5.4: pdfs of $T(n, 1)$: (a) analytical result, where the pdf of T is $Tr[1, 2]$, and experimental service time for data of size 3200 Kbytes, where (b) no other loads on the servers, and (c) other random loads on the servers

the service time of the best client-server pair as can be seen in the experimental results. In this case, the pdf of T_{min} is very close to that of T_1. To make the experimental results more interesting, some random loads are added to each server, so that the variance among the three client-server pairs is less than the variance within each pair, i.e., each pair behaves more similarly. The service times of three individual pairs (T_1, T_2, and T_3) and the whole service time (T_{min}) are shown in Figure 5.4(c). Of those four pdfs (T_{min}, T_1, T_2 *and* T_3), that of T_{min} is the leftmost, which supports the analytical properties of $T(n, k)$ and the pdf model of T.

5.4 Design An Efficient System

The performance of a server system with redundancy can be improved in two dimensions: (1) given the total number of the servers in the system, data can be distributed wisely among the servers so that the overall service time servers is minimized. This is the data distribution problem, which consists of determining k and choosing proper MDS array codes accordingly; (2) once the data distribution is set, data should be read from servers wisely so that the whole service time is minimized. This is the data acquisition problem.

The data distribution problem is at the servers' side, while the data acquisition problem is at the client's side. Because of the properties of $E[T(n, k)]$, the two problems are uncorrelated, thus they can be dealt with separately.

5.4.1 Data Distribution Scheme

In a server system, with a given total number of servers, n, we need to determine the number k of the servers which store the *raw* data in order to maximize the performance of the whole system (i.e., to minimize the mean service time of client's data request); given k, the rest of the servers can store the *redundant* data. When n and the pdf $f(t)$ are fixed, $E[T(n, k)]$ decreases monotonically as k decreases. This means that in order to make $E[T(n, k)]$ small, k should be as small as possible. On the other hand, however, the smaller k is, the more data needs to be stored on each server, since the total amount of the data a client needs is always fixed; this means higher service time from each server. Our goal is to find such a k that when both sides of the problem are considered, $E[T(n, k)]$ is minimized.

After the parameter k is determined, in order to achieve optimal performance in terms of $E[T(n,k)]$, we can use MDS array codes to distribute the redundant data so that data from *any* k servers can be assembled to form the whole of the requested data, as was shown in the introduction. Now the remaining problem is to determine k to minimize $E[T(n,k)]$. Applying the pdf model of each server's service time, T, and using MDS codes for distributing the redundant data, we get that if the pdf of T is $Tr[a,b]$ when k=1, then for general k, the corresponding pdf is $Tr[\frac{a}{k},\frac{b}{k}]$, since the base width of the pdf is proportional to the data size. Theoretically, the optimal k can be calculated as follows:

$$k_{min} = argmin_k \int_0^\infty k \binom{n}{k} F(t)^{k-1}[1-F(t)]^{n-k} t f(t) dt \tag{5.7}$$

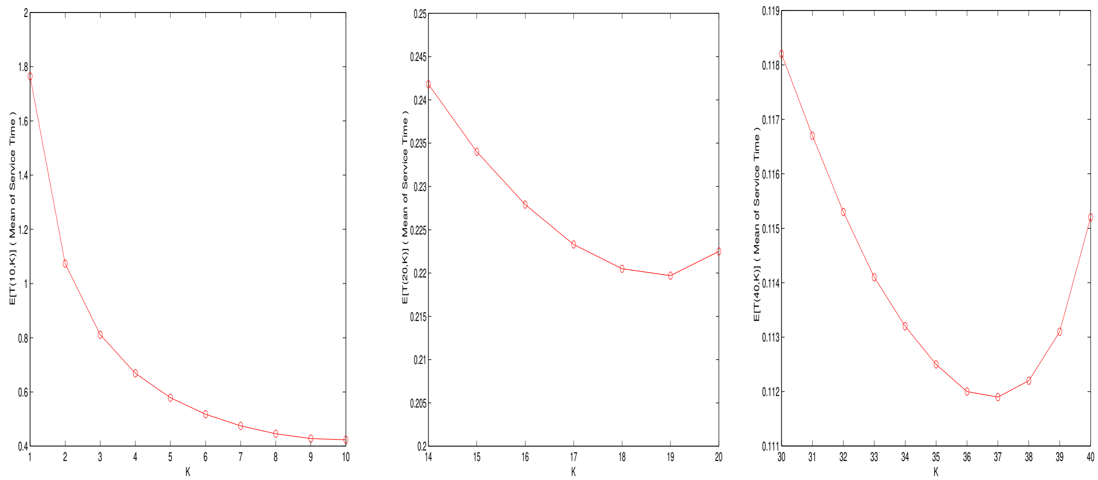
where $f(t)$ and $F(t)$ are as in Eq.(5.5) and Eq.(5.6), except that a and b should be replaced by $\frac{a}{k}$ and $\frac{b}{k}$ respectively. Notice that k_{min} is a function of the entire pdf $f(t)$, not only the mean $E(T)$ and the variance $Var[T]$.

Even for a simple pdf such as $Tr[a,b]$, the above equation can not be solved in closed form. But in practice, the system parameters a and b can be determined by experiments, then the above equation can be solved numerically. Figure 5.5 gives several examples of solving the above equation. In the examples, $a=1$ and $b=5$. For $n=10$, 20, and 40, $E[T(n,k)]$ is calculated for $1 \leq k \leq n$. The results are shown in Figure5.5(a)(b)(c), where (b) and (c) only show the last few values for k, since for small k, $E[T(n,k)]$ decreases monotonically as k increases. From the results, we can see (a) $k_{min}=10$, when $n=10$, (b) $k_{min}=19$, when $n=20$, and (c) $k_{min}=37$, when n = 40.

Even though the above examples use specific pdfs, the same method also apply with other pdfs by plugging suitable $f(t)$ into Eq.(5.7). Thus, for a given server system, such a k_{min} can always be found. Proper MDS array codes can then be used based on the (n,k) pair. Thus we get an optimal data distribution scheme for a given server system.

5.4.2 Data Acquisition Scheme

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 (or read) 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. For a specific distribution scheme, the client



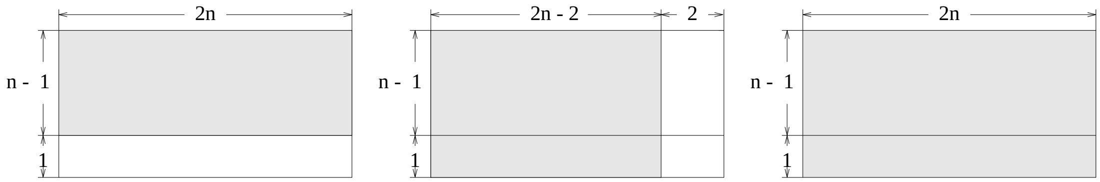
$(a)\ n = 10$ $(b)\ n = 20$ $(c)\ n = 40$

Figure 5.5: $E[T(n,k)]$ vs. k for different n, where $a = 1$ and $b = 5$

needs to calculate the pdfs of all possible data read schemes, and then choose an optimal read scheme. Since the read schemes are closely related to the MDS array code being used, here we will give an example using a specific code to show the guidelines for choosing an optimal read scheme.

In this example, the server system has $2n$ servers, and the data that the client requests can be assembled from any $2n-2$ servers, i.e., this is a $(2n, 2n-2)$ system. The B-Code in Chapter 3 can be used to implement this system. The data distribution using the B-Code is as follows: (1) the whole raw (*information*) data is partitioned into $2n(n-1)$ blocks of equal size (some paddings are added if necessary); (2) each of the $2n$ servers stores $n-1$ blocks of the data; (3) $2n$ blocks of redundant (or *parity*) data are calculated according to the encoding rules of the B-Code, i.e., each parity block is an XOR of suitable $2n-2$ raw data blocks, and then each server stores 1 parity block. The structure of the B-Code is shown in Figure 3.2 in Chapter 3.

The MDS property of the B-Code gives 3 schemes for reconstructing the whole raw data from the data stored on $2n$ servers, each of which has $n-1$ blocks of raw data and 1 block of parity data: (1) read from all of the $2n$ servers, each of which sends its $n-1$ blocks of raw data; (2) read from any $2n-2$ servers, each of which sends all of its n blocks of data (including raw and parity data); (3) read from all of the $2n$ servers, each of which sends all of its n blocks of data. The 3 schemes are shown in Figure 5.6, where the shaded parts are the data to read.



(*a*) *Scheme* 1 (*b*) *Scheme* 2 (*c*) *Scheme* 3

Figure 5.6: Three read schemes using the B-Code

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, then the client only needs to receive data from any $2n-2$ of the $2n$ requested servers. Let $E[T(2n,2n)]_{n-1}$, $E[T(2n-2,2n-2)]_n$ and $E[T(2n,2n-2)]_n$ denote the mean data service time of the three schemes respectively. From Property 1 of Theorem 5.1, $E[T(2n-2,2n-2)]_n > E[T(2n,2n-2)]_n$. But the relation between $E[T(2n,2n)]_{n-1}$ and either $E[T(2n-2,2n-2)]_n$ or $E[T(2n,2n-2)]_n$ is not so obvious, since in scheme (1) the client needs to wait for more servers, but needs less data (thus less service time) from each server. So to determine which scheme is best scheme for a given system, we need to calculate the pdf of the whole service time for all possible the schemes, which are scheme (1) and scheme (3) in this case.

Assume that the pdf of the time T for each server to send n blocks of data to the client is $Tr[a,b]$; then the pdf of T in scheme (1) is $Tr[\frac{n-1}{n}a, \frac{n-1}{n}b]$, since each server only needs to send $n-1$ blocks of data, and the pdf of T in scheme (2) or (3) is $Tr[a,b]$. Now the pdfs of the whole service time in the different schemes can be calculated according to Eq.(5.2), Eq.(5.5) and Eq.(5.6). Figure 5.7 shows the pdfs for different values of n, where $a=1$ and $b=10$.

Using Eq.(5.3), the mean of the whole service time of different schemes can be calculated. These means are listed in Table 5.1, for $a=1$ and $b=10$.

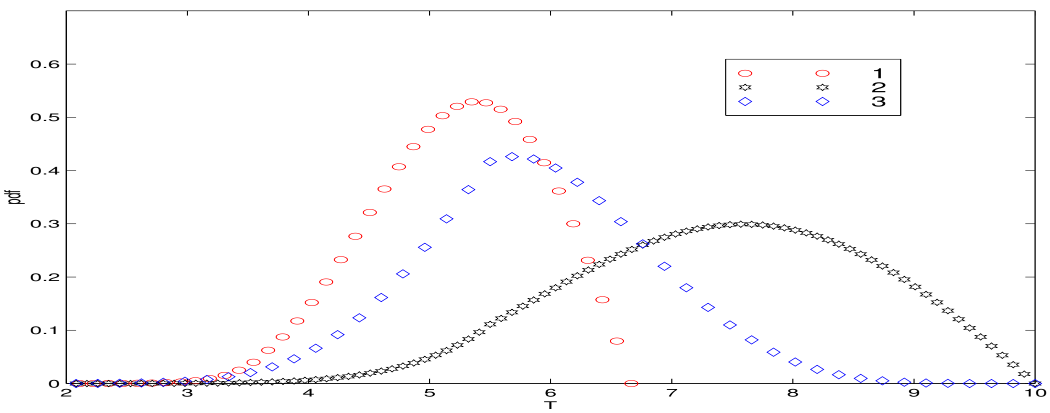
The above calculations show that the performance of the three schemes depends on the system parameter n (when a and b are fixed). In a small server system, scheme (1) is the best. As n increases, scheme (3) becomes better. For a system of 6 servers ($n=3$), scheme (1) is the best, but for systems of 14 servers ($n=7$) and 20 servers ($n=10$), scheme (3) is

n	3	7	10
$E[T(2n,2n)]_{n-1}$	**5.2195**	7.3128	7.8857
$E[T(2n-2,2n-2)]_n$	7.4089	8.4207	8.6976
$E[T(2n,2n-2)]_n$	5.8910	**7.2466**	**7.6786**

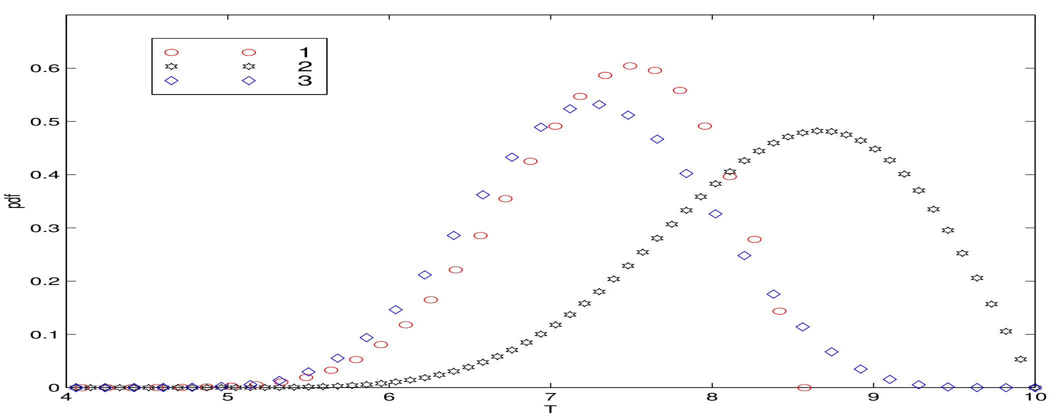
Table 5.1: Mean service time of different data read schemes, where $a = 1$, and $b = 10$

the best.

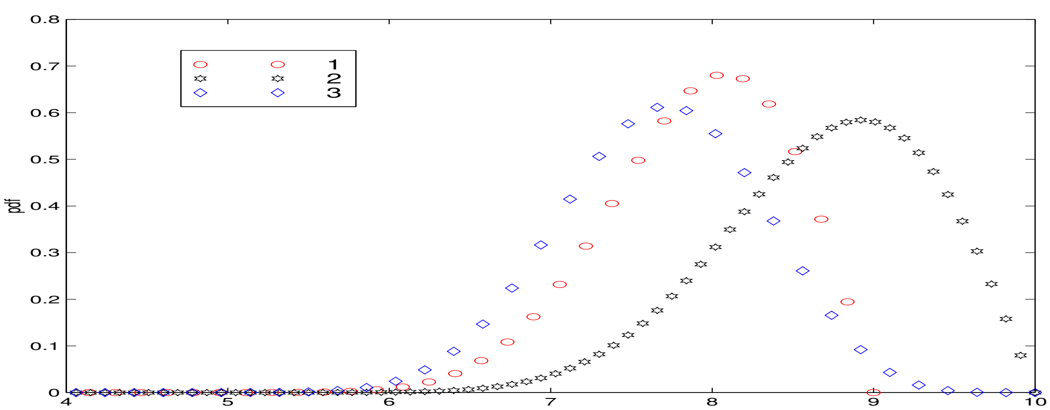
Though quite simple, the above example shows that after the data distribution is set at the server side, the client has different ways of reading data from the servers. For a given system (i.e., a certain the pdf of T, a fixed (n, k) pair and a particular code), there always exists an optimal read scheme for the client. Finding this scheme requires careful calculation. Since the read schemes are highly related to the codes used, exploring codes that offer more read choices is an interesting research problem.



(a) $n = 3$



(b) $n = 7$



(c) $n = 10$

Figure 5.7: PDFs of different data read scheme, where $a = 1$, $b = 10$; 1, 2 and 3 represent scheme (1), (2) and (3) respectively.

5.5 Summary

We have explored options for improving the performance of data server systems by introducing data redundancy based on array codes. We have proposed methods of achieving better performance at both the server side and the client side of systems, namely finding an optimal data distribution scheme and an optimal data acquisition scheme. We have also given a simple probability model for a real data server system, based on experimental results. Our additional experimental results also verify qualitatively our analysis results.

In this chapter, the system model is rather simple: data requests from one client or multiple clients are processed sequentially, i.e., at one time, and each server only handles one data request. But in a more practical system, such as a web server system, multiple data requests can come according to some stochastic process, and they can be processed by the servers in a parallel fashion such that the number of requests processed per unit time is maximized. Optimizing such systems is an interesting and useful research problem that might require that the servers use some sophisticated scheduling schemes, based on the results in this chapter.