On Correlated Failures in Survivable Storage Systems

Mehmet Bakkaloglu, Jay J. Wylie, Chenxi Wang, Gregory R. Ganger

May 2002

CMU-CS- 02-129

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

{mehmetb, jwylie, chenxi, ganger}@andrew.cmu.edu

Abstract

The design of survivable storage systems involves inherent trade-offs among properties such as performance, security, and availability. A toolbox of simple and accurate models of these properties allows a designer to make informed decisions. This report focuses on availability modeling. We describe two ways of extending the classic model of availability with a single "correlation parameter" to accommodate correlated failures. We evaluate the efficacy of the models by comparing their results with real measurements. We also show the use of the models as design decision tools: we analyze the effects of availability and correlation on the ordering of data distribution schemes and we investigate the placement of related files.

Acknowledgements: This work is partially funded by DARPA/ITO's Organically Assured and Survivable Information Systems (OASIS) Program (Air Force contract number F30602-99-2-0539-AFRL). We thank the members and companies of the PDL Consortium (including EMC, HP, Hitachi, IBM, Intel, Network Appliance, Panasas, Seagate, Sun and Veritas) for their interest, insights and support. We also thank Microsoft Corporation for sharing their availability measurements of the desktops in their campus.

Keywords: Survivable Storage Systems, Availability Modeling, Storage Nodes, Correlated Failures.

1 Introduction

Survivable storage systems [Wylie2000] encode and distribute data over multiple storage nodes to survive failures and malicious attacks. PASIS is one such system that is configurable to support many data distribution schemes, including threshold schemes [Shamir1979], for improving storage availability and security. There are many other projects, such as Farsite [Bolosky2000] and OceanStore [Kubiatowicz2000], which use similar techniques to achieve similar goals. A common property of these techniques is that they have parameters that affect the properties of the system. Setting these parameters in an ad-hoc fashion may be sub-optimal in terms of meeting users' requirements. In [Wylie2001], we describe a framework to select these parameters (i.e., a scheme) in the context of performance, security, and availability trade-offs. In general, there are two ways of quantifying these three properties: measurements and models. Models can be parameterized, allowing one to explore a wide range of systems quickly. In this report, we focus on availability modeling.

The classic approach to modeling the availability of distributed storage is to assume that storage nodes have identical availabilities and independent failures. This model is useful because it is simple and it enables a system designer to reason about the effect of average node availability on overall system availability. However, it ignores correlations known to exist among storage node failures [Amir1996].

The most accurate way of modeling correlated failures is to specify exactly the probability of each subset of nodes being unavailable. But, such a model is too complex in practice and provides little intuition to system designers. A better approach would be to extend the classic model with a single correlation parameter that indicates the level of failure correlation in the system. In this report, two such models are described. The first one is based on conditional probabilities, and the second one is based on the Beta-Binomial distribution. Both of these models have two parameters that describe the system: *average storage node availability* and *correlation level*.

An interesting issue in selecting a scheme, for a given system, is the ordering of the schemes based on their availabilities. It may be the case that the user specifies the properties of the underlying system incorrectly. In such a case, the relative ordering of the schemes can be affected and ultimately it may result in the selection of a sub-optimal scheme. In this report, we investigate how the ordering of the schemes is affected with respect to average storage node availability and correlation level.

The problem of optimal file placement in a survivable storage system has been studied before [Douceur2001b]. In those studies, it is generally assumed that files are independent from one another. In reality, there could be dependencies between files and directories. For example, to access a file, the directories on the path to the file have to be traversed. If the directories are not available, the file will not be available. Thus, the relative placement of related files and directories may impact the availability of the overall system. In this report, we analyze cases to show where it is important to consider related files and directories in doing placement.

This report is organized as follows. Section 2 overviews the different data distribution schemes. Section 3 discusses the importance of selecting the correct data distribution scheme for a system and briefly explains a framework for doing this selection. Section 4 discusses modeling the availability of data distribution schemes and describes two approaches that are able to capture the correlation between failures of storage nodes. Section 5 analyzes the effect of availability and correlation on the ordering of schemes. Section 6 investigates the importance of considering related files in doing file placement. Section 7 summarizes the contributions.

2 Data Distribution Schemes

This section describes data distribution schemes from the literature that are used in survivable storage projects.

Threshold Schemes. Threshold schemes are a way of encoding data. A threshold scheme has three parameters: *n*, *m* and *p* (where $n \geq m \geq p > 0$). Data is divided into *n* shares, of which any *m* is sufficient to fully recover the data. Less than *p* shares give absolutely no information about the data. The formation of the shares is called encoding and the process of combining the shares to reconstruct the original data is called decoding. An example scheme is replication. Each replica is the original data, thus, $m=1$ and $p=1$. Other threshold schemes include Information Dispersal [Rabin1989], Secret Sharing [Shamir1979], Ramp Schemes [Blakley1985], Read-Solomon Codes [Plank1997] and Tornado Codes [Luby1998]. Table 1 lists a number of well-known threshold schemes.

n-m-p	**Name**
n-1-1	Replication
n-n-1	Decimation (Striping)
n-n-n	Splitting (XOR-ing)
n-m-1	Information Dispersal
n-m-m	Secret Sharing
n-m-p	Ramp Schemes

Table 1. Threshold Schemes

In general, the *n* shares formed during encoding are stored on *n* different storage nodes. For a file to be available, at least *m* of the nodes have to be available. For most of the schemes, storing the *n* shares on different nodes increases availability. This is because the system can tolerate up to (*n-m*) node failures. Additionally, for an adversary to steal the data in its entirety, she must compromise at least *m* storage nodes. Security can be further enhanced, by storing shares on nodes with different operating systems. Thus, a weakness in one operating system may not lead to the compromise of nodes with different operating systems. PASIS [Wylie2000] and Farsite [Bolosky2000] are two example storage systems that use threshold schemes. In OceanStore [Kubiatowicz2000] and in Intermemory [Goldberg1998] Read-Solomon Codes and Tornado Codes are used.

Quorum Systems. In some data replication protocols, quorum systems are employed. A quorum system is a set of subsets of U (the universe of servers), of which every pair of sets intersect. A read/write operation is done by first selecting a quorum and then by accessing all of the elements in that quorum. Since every pair of quorums intersect, the user always has a consistent view of the system [Amir1998]. Optimal quorums for masking *b* Byzantine failures, where the intersections contain at least $2b+1$ servers are described in [Malkhi2000].

The common idea in these different data distribution schemes is that they divide a piece of data into a number of shares and only a subset of the shares are required to fully reconstruct the original data.

3 Selecting the Correct Scheme

Parameter options for threshold schemes (i.e., values of *n*, *m* and *p*) create a large space of possibilities. Each scheme has different properties, thus for a given set of requirements the optimal scheme for different systems may be different. In order to select the optimal scheme, we first have to be able to compare the schemes and to compare the schemes we need to be able to quantify them. In quantifying the schemes, three metrics are used: performance, confidentiality and availability. Here, we give an overview of each metric. The details of the model for each one can be found in [Wylie2001].

Performance. Performance is calculated in terms of throughput (MB per second). The time to produce the shares (i.e., encode the data), the number of shares, the size of each share and the number of shares required to reconstruct the data depend on a schemes' parameters *n*, *m* and *p*. In [Wylie2001], we use a simple model to compare the performance of different schemes. Specifically, we examine the effects of changing the workload, the CPU speed at the client side, the network bandwidth and the storage node response times.

Confidentiality. Each scheme offers a different level of confidentiality. This is due to the three parameters *n*, *m* and *p*. As *m* increases, the attacker has to steal more shares to fully reconstruct the file and less than *p* shares give absolutely no information to the attacker. So higher *m* and higher *p* mean higher confidentiality. On the other hand, having more shares (i.e., higher *n*) degrades confidentiality since there will be more vulnerable nodes. The unit used for confidentiality is the effort required by the adversary to gain information.

Availability. Availability is defined as the probability that a file can be accessed at any given time. With threshold schemes, files are encoded into *n* shares, of which *m* or more are sufficient to fully reconstruct the file. In [Wylie2001], we assume the failures of storage nodes are independent and we use the classic availability model [Siewiorek1992]. For reads the model is,

$$f(n, m, Avg.Avail) = \sum_{i=m}^{n} \binom{n}{i} (Avg.Avail)^{i} \quad (1 - Avg.Avail)^{n-i}$$

Note that for writes, the availability can be defined in several different ways. One extreme case is to require any *m* nodes to be available and the other extreme case is to require *n* specific nodes to be available. In the former case, read availability gets affected. The unit of availability is nines (i.e. 0.99 means 2 nines).

3.1 The 3D Trade-off Space

The section explains how scheme selection is done. We focus on seven classes of schemes: replication, replication with cryptography, ramp schemes, information dispersal algorithms, secret sharing, short secret sharing and splitting. Each scheme is quantified in terms of the three metrics that are explained above. In Figure 1, each scheme is represented with a different tone of gray. The graph shows the best performing scheme and its performance for a given confidentiality and availability level. Note that the possible schemes cover only a portion of the entire trade-off space. The reason is that there exists no scheme that can provide maximum confidentiality, maximum availability and maximum performance at the same time. Thus, a user must make trade-offs among the three properties. In order to select a scheme using the 3D Trade-off space, the user specifies a confidentiality and availability level in accordance with her requirements and then uses the graph to identify the optimal scheme in terms of performance.

Default Configuration. The default configuration we use in [Wylie2001] consists of 10 storage nodes, 100Mb/sec network bandwidth, 32KB files (blocks), a 0.5 read/write ratio, 0.995 storage node availability and the effort required to circumvent cryptography (e.g., by stealing the key) is equal to the effort required to break into a storage node. Figure 1, shows the resulting graph with respect to the seven schemes.

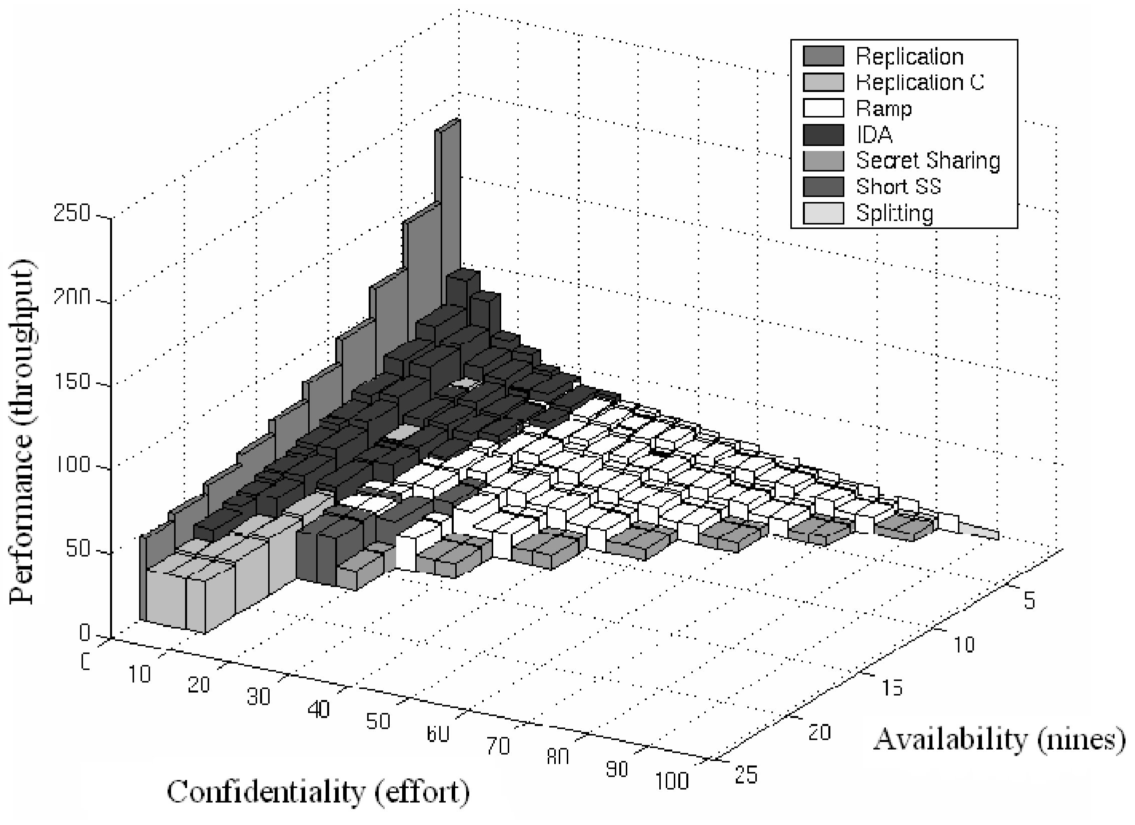

Figure 1. Default Configuration

Different Configurations. In analyzing the effects of different system configurations we use the default configuration as the base case. In [Wylie2001], we examine the effects of different system configurations such as higher network bandwidth, different read/write ratios in the workload, effort of cryptanalysis being greater than effort required to break into a storage node, and lower storage node availability. The detailed description of the first three is beyond the scope of this paper. Figure 2 shows the effect of having lower storage node availability.

An inaccurate estimate of average storage node availability can drastically affect the space that is covered by the schemes. For example, if the storage node availability is estimated as 0.95, as opposed to 0.995, the original graph (i.e., Figure 1) gets contracted and covers a smaller portion of the space. This causes the user to become unaware of the fact that there are schemes that provide higher levels of availability. Furthermore, if the user selects a scheme using the contracted graph, the selected scheme would have a lower performance than the scheme that would have been selected, if the correct graph had been used. These imply that the optimal scheme, for a given set of requirements, heavily depends on individual storage node availabilities. Also an interesting phenomenon is the change in the ordering of the schemes as average availability changes. A detailed analysis of the ordering of the schemes is given in Section 5.

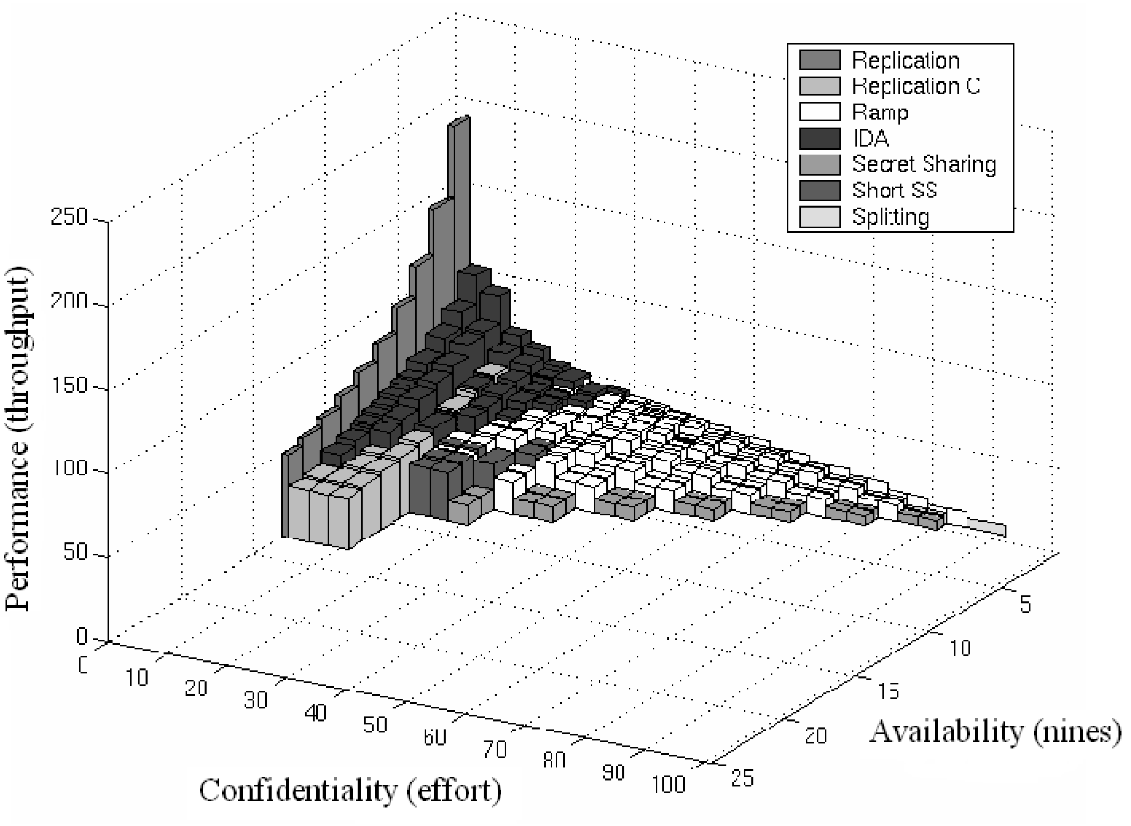

Figure 2. Lower Storage Node Availability

3.2 Discussion

As discussed in the previous section, using simple models for performance, confidentiality and availability enables us to explore the trade-off space among the three properties. Different configuration parameters result in different selection surfaces. Another important point is that each model has its inherent assumptions that affect the 3D graph. For example, in the availability model it is assumed that storage node failures are independent. In the next section, we analyze the availability model in more depth.

4 Modeling the Availability of Threshold Schemes

In this section, we investigate modeling the availability of threshold schemes. The ideas here can be applied to modeling the availability of other data distribution schemes. First, we describe the basics and analyze the underlying assumptions of the classic way of modeling availability. In Section 4.3, we explain two approaches to correlation-aware availability modeling and we compare the two models to real measurements. In Section 4.5, we describe related work.

In modeling the availability of threshold schemes, we treat the individual storage nodes as black boxes in the sense that we only know when the node is up or down and we have no knowledge of why this has happened. We assume that only the client link failures are detectable and these are the only failures that are not counted as the server being unavailable. We model the availability of a scheme using the availability values of the black boxes and we do not have a clear picture of the relations between black boxes. More specifically, we have no knowledge of the underlying network and the different administrative domains.

4.1 Availability of Threshold Schemes

The availability of a storage node is defined to be the percentage of time the node is able to service requests. One common way of expressing it is as nines of availability (e.g., 0.99 is 2 nines) [Gray1991]. There are two kinds of availabilities for threshold schemes: read and write availability. A file is available to being read if there are *m* or more up-to-date shares that are available at the time of read. On the other hand, there are several ways that write availability could be defined,

- Any *n* of the *N* storage nodes in the system being available (e.g., creating a file, rewriting a file)
- Any *m* of the *N* storage nodes in the system being available (e.g., creating a file, rewriting a file)
- All *n* of *n* specific nodes being available (e.g., updating a file)
- Any *m* of *n* specific nodes being available (e.g., updating a file)

Clearly, the way writes are done affects the availability of reads. In our model, we focus on read availability; it is straightforward to extend our model to write availability.

4.2 The Classic Model

In the literature [Siewiorek1992], a common way of modeling the availability of a given set of storage nodes is by assuming:

- The failures of the storage nodes are independent and;
- The storage nodes have identical availabilities that are equal to the average availability.

The formula used in this case is:

$$f(n, m, Avg.Avail) \quad = \quad \sum_{i=m}^{n} \binom{n}{i} \quad (Avg.Avail)^{i} \quad (1 - Avg.Avail)^{n-i}$$

Independence Assumption. To see how the independence assumption of the classic model affects the outcome, consider the following example shown in Figure 3. Assume two machines each with availability of 0.99. The classic model's estimation for n=2 and m=1 is 0.9999 (4 nines). But, in reality, the availability of this scheme could be anywhere between 0.99 and 1.0. If the downtimes of the two nodes completely overlap, then replication provides no benefit. On the other hand if the downtimes do not overlap at all then the availability of the scheme is 1.00.

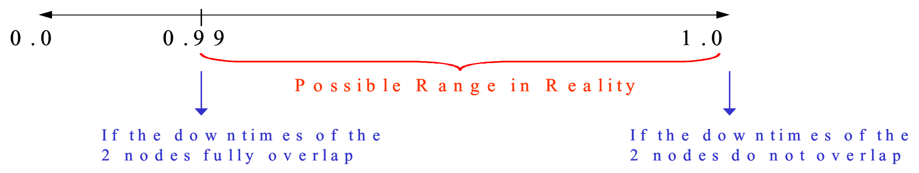
0.0
0.99
1.0
Possible Range in Reality
If the downtimes of the 2 nodes fully overlap
If the downtimes of the 2 nodes do not overlap

Figure 3. Availability of scheme with n=2 and m=1

In the following sections, we show that, by relaxing the independence assumption, the accuracy of the model can be improved substantially.

4.3 Correlation-Aware Availability Modeling

4.3.1 Requirements

The classic model shown in the previous section is an easy tool to use. It is a function of the form,

$$f\,(\textit{Avg. Avail., n, m})$$

As explained in Section 4.2 the independence assumption of the classic model may lead to inaccurate estimations. The definition of independence is,

$$P\,(\text{Node X down} \mid \text{States of other nodes in the system}) = P\,(\text{Node X down})$$

The independence assumption is not always valid. Several studies show that storage node failures are correlated. In [Bolosky2000][Douceur2001c], the actual scheme availabilities (replication in their case) differ from the calculations that assume independent failures. In [Amir1996], they analyze 14 machines in a wide area network and report that crashes are correlated. Our measurements of web server availabilities also support the fact that,

$$P\,(\text{Node X down} \mid \text{Node Y down}) > P\,(\text{Node X down})$$

Therefore, the new model should have the following properties:

- It should be a tool that is useful to system designers, which means that it should not have too many parameters.
- It should be more accurate than the classic model for cases when there are correlations between failures.
- While achieving the first two it should not hurt cases when the failures are actually independent.

In achieving these requirements, we investigate the possibility of relaxing the independent failures assumption without overly complicating the model. More specifically, the required function is of the form,

$$f\,(\textit{Avg. Avail., n, m, Correlation Level})$$

In the following sections, we describe our datasets and two approaches to correlation-aware availability modeling. The first approach is based on conditional probabilities. The second approach is based on the Beta-Binomial distribution. The Beta-Binomial distribution has been used previously to model correlated events such as the number of infectious diseases in a household [Griffiths1973] and failures in multi-version software [Nicola1990].

4.3.2 Datasets Used

In our work we use two datasets. The first one is our availability measurements of Web Servers and the second one is measurements of Desktops [Bolosky2000].

Web Servers. This dataset consists of our measurements of more than 100 web servers from September 2001 to December 2001. The web servers come from various domains such as com, edu and gov, located in various countries. The client was an Intel 600MHz Linux machine. The data was collected, by reading a file off each web server every 10 minutes. If the server responded and if the file was valid (e.g., not a HTTP 404 Error and has HTTP OK in its header), then we recorded this as the server being available. In this sense, the availability measured is the service availability. In our experiments, we exclude some of the servers due to reasons such as protocol version incompatibilities. We use 99 of the servers, which are listed in the appendix. The failures between the web servers were found to be correlated. In order to have more cases to compare our models against, in addition to using the entire dataset, we also use subsets of the entire dataset. In this report, we show results from 4 representative datasets. Dataset-1, Dataset-3 and Dataset-4 are subsets taken by time and Dataset-2 is the entire dataset. Dataset-1 is weeks [1-3], Dataset-3 is weeks [8-10] and Dataset-4 is week 8.

Desktops. This dataset consists of anonymized desktop availability statistics measured at Microsoft Corporation [Bolosky2000]. Machines, in the campus, were pinged hourly from July 1999 to August 1999. The failures between the machines in this set were almost independent. Here, again to have more cases to compare our models against, we analyzed subsets of the entire dataset. In this dataset, there are only 757 valid samples for each machine. Thus, rather than analyzing different periods of the entire dataset, we analyze different subsets of machines. In this report, we show results from 4 representative subsets where each one is constructed using a subset of the machines. Dataset-1 is machines [1-2000], Dataset-2 is machines [1001-3000], Dataset-3 is machines [2001-4000] and Dataset-4 is machines [4001-6000].

4.3.3 The Model Based on Conditional Probabilities

Calculating Correlation

The most comprehensive way of determining correlation is to calculate the probability of each subset of nodes being unavailable. But, such a model is a difficult tool to reason about the effects of correlation on the availability of different schemes. A simpler approach would be to use a single correlation parameter in the model. Therefore, the question is "How can such a parameter be measured?" According to the classical definition of correlation, two variables have a correlation close to 1, if they mostly take on the same value and they have a correlation close to –1, if they mostly take on opposite values. The problem with using this approach for our case is that, if two machines are up 99% of the time, they will have a high correlation value regardless of how correlated their failures are. Therefore, this way of calculating correlation is not suitable for nodes with high availabilities.

A better approach is to calculate the average conditional probability. A similar approach was used in modeling the correlated failures in multi-computer systems [Tang1992]. Average conditional probability is calculated as follows.

$\forall$ Pair of nodes (X, Y) in the system, where $X \neq Y$,

Calculate P (Node X is down | Node Y is down)

The average of all these probabilities is the correlation level of the system.

[Note that for two nodes A and B, P (Node A down | Node B down) is not necessarily equal to P (Node B down | Node A down).]

When the correlation level is 0, this means that there is absolutely no overlap between the down times of storage nodes (note that 0 does not mean independence). When it is 1, it means that all of the down times are overlapping.

Another way of calculating the correlation level of a system could be by finding,

($\forall$ X, Y Avg. P (Nodes X&Y are down)) / ($\forall$ Z Avg. P (Node Z is down))

Note that this value is not necessarily equal to the average conditional probability. This is due to the fact that, in calculating the two values, averages are taken in different places. Recall that,

Avg. Conditional Probability = Avg. (P (Node X is down | Node Y is down))

= Avg. (P (Nodes X&Y are down) / P (Y is down))

We propose that either one of these can be used as the correlation level. A heuristic is to use the maximum of the two values as the correlation level.

The correlation level is an indication of 2-way correlations. The remaining question is how to determine higher-levels of correlations: the values of 3, 4, … of the storage nodes being correlated. The correlation level, as it is defined here, does not determine these values. The example in Table 2 illuminates this fact. In the example, there are two cases regarding the states of nodes A, B and C. In both of the cases, the average conditional probability (i.e., the correlation level as defined above) is 0.5. But, in the first case, the probability that the three nodes are down simultaneously is greater than zero while in the second case this probability is equal to zero.

[Notation: U- up D- down, assume a period of 6 time units, so in case 1 node A is, up-up-down-down-up-up]	
Case 1	Case 2
A UUDDUU	A UUDDUU
B UUDUDU	B UUDUDU
C UUDUUD	C UUUDDU

Table 2. Example – Difference in P (Node A down | Node B and C down)

The new model has an empirical value for P (A node is down | Another node is down) (i.e., the correlation level), but it does not have a value for P (A node is down | Two other nodes are down). To determine the availability of schemes with $n>2$, the model has to estimate the higher order conditional probabilities. The next section describes the model based on conditional probabilities in more detail and explains how this estimation is done.

The Model

The availability model we suggest has 4 parameters: n, m, *average availability* and the *correlation level*. As an example consider the tree-like structure shown in Figure 4. This structure is an easy way of viewing conditional probabilities. We number the nodes from 1 to N, but note that the model assumes that all of the storage nodes are identical.

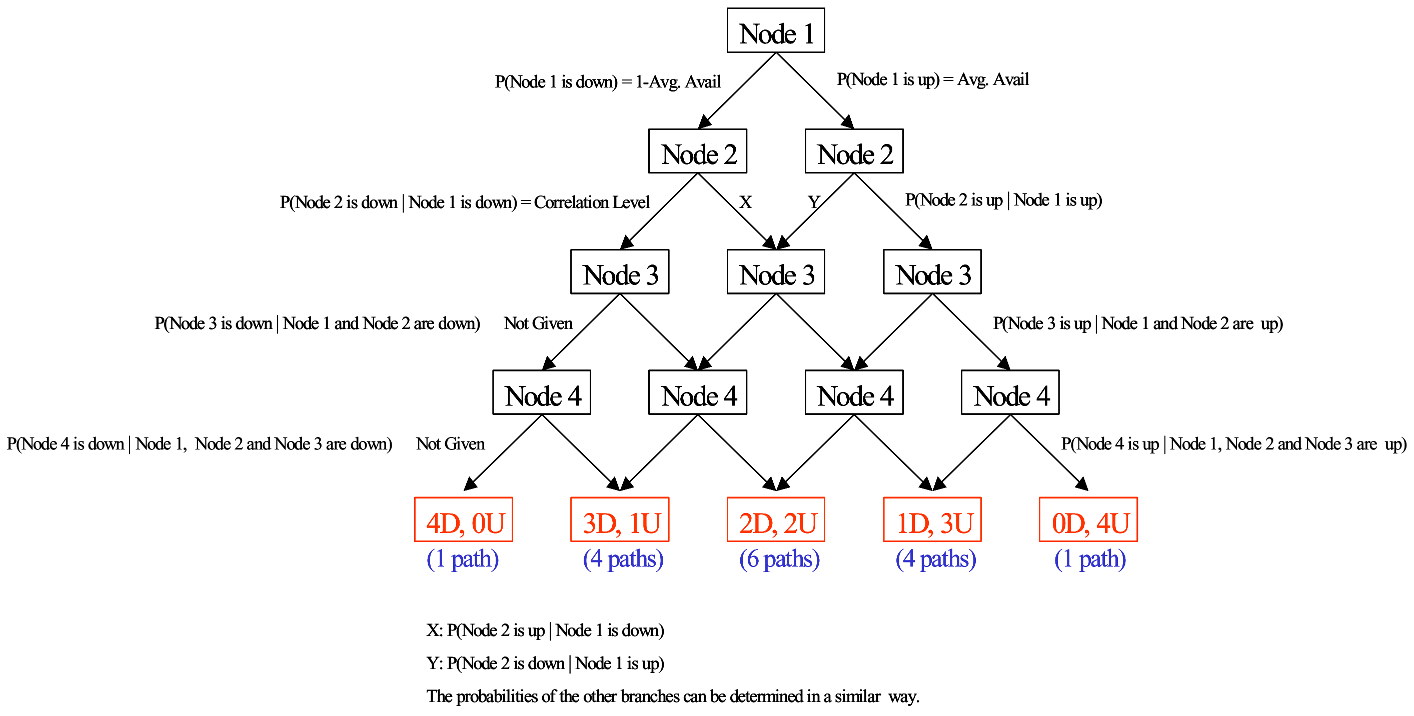
Node 1
P(Node 1 is down) = 1-Avg. Avail
P(Node 1 is up) = Avg. Avail
Node 2
Node 2
P(Node 2 is down | Node 1 is down) = Correlation Level
X
Y
P(Node 2 is up | Node 1 is up)
Node 3
Node 3
Node 3
P(Node 3 is down | Node 1 and Node 2 are down)
Not Given
P(Node 3 is up | Node 1 and Node 2 are up)
Node 4
Node 4
Node 4
Node 4
P(Node 4 is down | Node 1, Node 2 and Node 3 are down)
Not Given
P(Node 4 is up | Node 1, Node 2 and Node 3 are up)
4D, 0U (1 path)
3D, 1U (4 paths)
2D, 2U (6 paths)
1D, 3U (4 paths)
0D, 4U (1 path)
X: P(Node 2 is up | Node 1 is down)
Y: P(Node 2 is down | Node 1 is up)
The probabilities of the other branches can be determined in a similar way.

Figure 4. The Availability Tree

Structure of the tree

In the tree, the left branch of each node is the probability that the node is unavailable given the path that has been traversed to reach that node. The probability on the right branch is the probability that the node is available. The leaf nodes represent all the possible cases given 4 nodes: 4 of them are down, 3 of them are down and 1 of them is up, etc. Since it is assumed that all the nodes are identical, all the paths leading to a specific leaf node have the same probability. The probability of a path is found by multiplying the probabilities of the branches along that path. To find the probability of a leaf node, it is sufficient to multiply the probability of any path leading to that leaf by the number of paths that lead to that leaf. Therefore, given a scheme, it is trivial to calculate the availability of the scheme by adding the probabilities of the leaf nodes that make that scheme available. For example, to find the availability of the scheme with n=2 and m=2 we do the following. The scheme is available when both of the nodes is up and this probability is equal to

P (Node 2 is up | Node 1 is up) * P (Node 1 is up)

Calculation of the Values of the Branches

In the model, we assume we are given the average availability and the correlation level. In the tree, the average availability is the value of the right branch of Node 1 and the value of the left branch is 1-(value of right branch). The correlation level refers to the left branch of the first Node 2. 1 minus this value is the value of the right branch (shown as x on the tree). Given the definition of the tree structure, this means that we can calculate P (Node 1 is down and Node 2 is up),

P (Node 1 is down & Node 2 is up) = x * P (Node 1 is down)

Also note that we can now calculate the value of left branch of the second Node 2 (shown as y). This is because we assume all the nodes are the same, which implies that,

P (Node 2 is down & Node 1 is up) = P (Node 1 is down & Node 2 is up)

And since we know the values of P (Node 1 is down and Node 2 is up) and P (Node 1 is up) we can calculate y using the following equation,

P (Node 2 is down & Node 1 is up) = y * P (Node 1 is up)

Now that we know the value of the left branch of the second Node 2, we can also find the value of its right branch. At this stage we have calculated all the values of the left and right branches of Node 1 and that of the two Node 2's.

But, we have no way of calculating the value of the left branch of the first Node 3. This is where estimation is needed. Note that the left branch of the first Node 3 is P (Node 3 is down | Node 1 and Node 2 are down).

Estimation

To make a good and generic estimation, we examined our datasets to determine if there were any patterns or relations between the parameters of the model and the values the model has to estimate. First, we introduce the notation:

$$R(k) \quad = \quad \frac{\{\ \forall X, \quad X \subseteq U, \quad |X| = k, \qquad Avg.(P(k\_nodes\_are\_down))\ \}}{\{\ \forall Y, \quad Y \subset U, \quad |Y| = k-1, \qquad Avg.(P([k-1]\_nodes\_are\_down))\ \}}$$

$$where \quad R(1) = 1 - Avg.Avail.$$

[U is the set of all nodes in the system.]

We calculated the empirical R(x) values, for x=1,2 … 10, for the datasets described in Section 4.4.2. They are shown as solid lines in Figures 5 and 6. Figure 5 shows the datasets from the Web Servers measurements. Figure 6 shows the datasets from the Desktops measurements. Note that the scales of the y-axes are different for the two figures. The dotted lines are the model's values. The *model*'s values are the probabilities shown on the leftmost branches of the tree,

$$\begin{aligned} R(1) &= P(Node1\_is\_down) \\ R(2) &= P(Node2\_is\_down \mid Node1\_is\_down) \\ R(3) &= P(Node3\_is\_down \mid Node1\_\&\_Node2\_are\_down) \\ etc. \end{aligned}$$

Note that, the equalities in the equations of R(2) and R(3) hold due to the assumption in the model that all the nodes are identical. The two input parameters, average availability (R(1)) and correlation level (R(2)), are marked for Dataset-1 shown in Figure 5. The model has to estimate all the other R(x) values (x=3, 4 … 10). The difference between R(2) and R(1) is an indication of the level of 2-way failure correlations. When this difference is zero there is no 2-way correlation and if this is the case then estimating that,

$$R(3) \quad = \quad R(2) \quad = \quad R(1)$$

would be reasonable. On the other hand, when this difference is positive, this means that failures are correlated.

Both of the datasets shown in Figures 5 and 6 (solid lines), in general, indicate that *R*(3) values are higher than *R*(2) values. For most of the lines it is also the case that *R*(x)'s are higher than *R*(x-1)'s. Now that we have identified the increase we have to determine how this increase happens. First of all, we want to make sure that we do not over-fit our datasets, therefore, our formula should be as generic as possible (i.e., we do not want a fourth or fifth order polynomial equation). We propose the following estimation,

$$R(x) \quad = \quad R(x-1) \quad + \quad \frac{[R(x-1)-R(x-2)]}{2} \qquad for \quad x = 3,4,\ldots 10$$

which says the difference between *R*(x) and *R*(x-1) decreases *exponentially*, as *x* increases. To ensure the estimated probability does not exceed 1, we add the following term to the previous formula,

$$R(x) \quad = \quad \min\Big(\; R(x-1) \quad + \quad \frac{[R(x-1)-R(x-2)]}{2} \quad , \quad \frac{R(x-1)+1}{2} \; \Big) \quad for \quad x = 3,4,\ldots 10$$

Figures 5 and 6 show that the estimations obtained (dotted lines), using the above formula, are lower than the actual values (solid lines). The lines that show the estimations tend to flatten out earlier. We describe some of our attempts to overcome this problem. In the discussion, we use *D*(x) to represent the difference between *R*(x) and *R*(x-1).

Our first attempt to correct the problem was to change the estimation in a way that *D*(x) decreases *linearly* after x>4. The problem with this approach is that for Dataset-1 shown in Figure 5 (Web Servers), the estimations become high compared to the real values. Also the estimations for the datasets, which we thought would benefit from this approach, did not improve much. For example, for the Desktops datasets shown in Figure 6, the problem is that since initially *D*(x) decreases exponentially, *D*(4) becomes very low. Thus, changing to a linear decrease, after x>4, does not provide much benefit. For Dataset-3 and Dataset-4 shown in Figure 5, the input parameters (i.e., *R*(1) and *R*(2)) are very low compared to *R*(x), where x≥6. Thus, capturing this behavior in the estimations is difficult.

Our second attempt to correct the flattening problem was to change only the way *R*(3) is estimated. The rationale here is to have a very accurate estimate for *R*(3) from which the further estimations can benefit. A way to do this is by estimating *R*(3) using:

$$R(3) \quad = \quad R(2) \quad + \quad \frac{[R(2)-R(1)]}{(4/3)}$$

This heuristic provides a slightly better estimation compared to the original method, but it lacks the generality of that method. Thus, it is possible that this results in over-fitting.

One final heuristic could be to use *R*(4) as the correlation level. The problem with this approach is that such a parameter is not easy to reason about and once again it could be that we are over-fitting our data. Note that we have access to only two datasets. Having more datasets would enable us to make a more generic estimation.

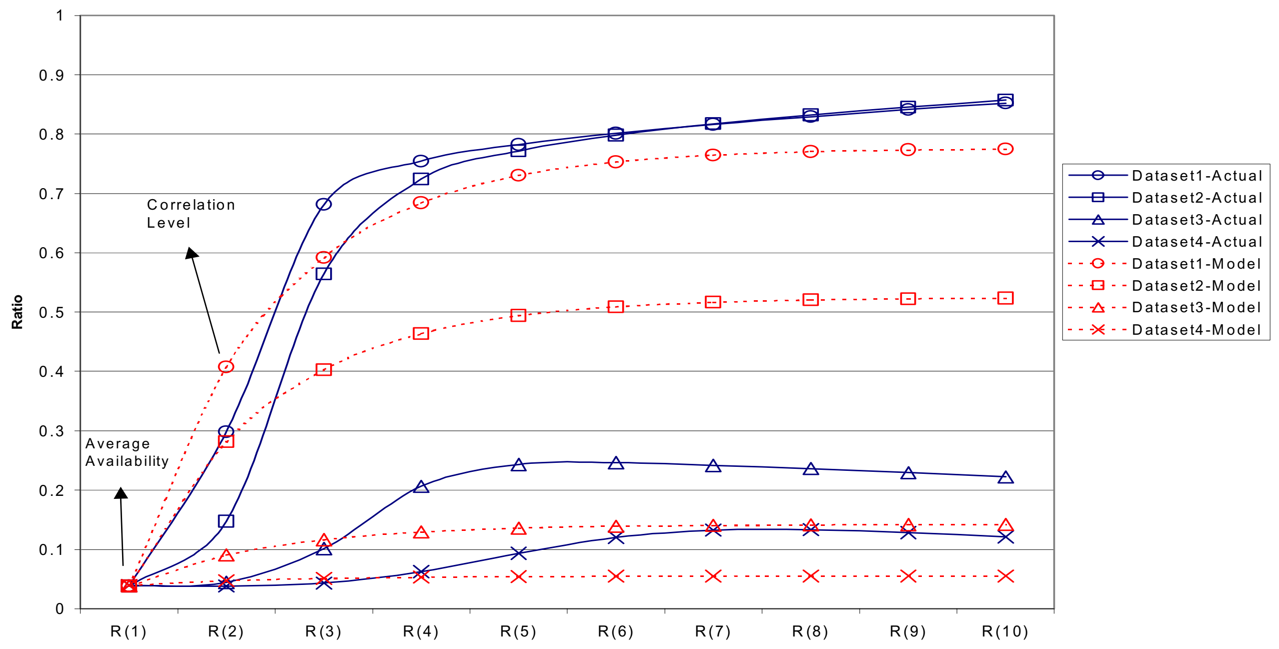

Figure 5. Each line represents a different subset of the entire Web Servers dataset. Each dotted line is the estimation of the model for the corresponding solid line (e.g., The dotted line with triangles corresponds to the solid line with triangles). The average availability and correlation level for Dataset1 is shown on the figure. R(x) is defined in the text.

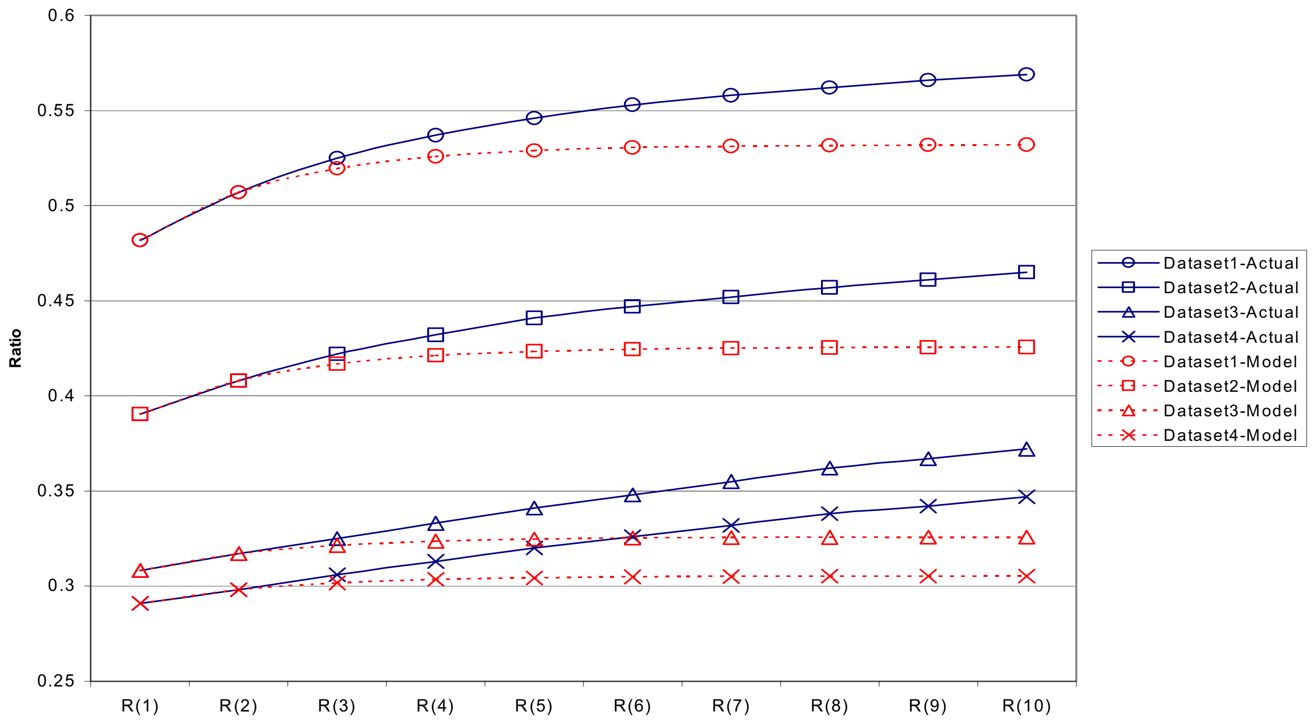

Figure 6. Same as the above figure except that the data here comes from the Desktops dataset. Note that the y-axis here is from 0.25 to 0.6.

Substituting the Estimation into the Tree

Now that we know how to calculate all the *R*(x)'s, we can substitute them into the tree and we can calculate the values of the branches of all the nodes (in a similar way we did for the branches of the Node 2's). Note that the model reduces to the classic model when Correlation Level = (1-Avg.Avail.).

An example

Using the tree let us calculate the availability of the scheme with *n*=4 and *m*=3. For this scheme to be available there has to be 3 or more nodes that are available. Therefore,

P (Avail.)= P (4 up) + P (3 up, 1 down)

Recall that every path from Node 1 to a leaf node is a possible scenario. The probability of a path is calculated by multiplying the probabilities on the branches leading to that leaf. The paths that lead to the same leaf node have the same probability. Thus, the probability of the leaf node can be calculated by multiplying the value of any path that leads to that node by the number of paths that lead to that node. For the case of P (3 up, 1 down) there are 4 paths that lead to the leaf node labeled [3U, 1D]. For P (4 up) there is only one path.

4.3.4 The Beta-Binomial Distribution

The second model we suggest for modeling the availability of correlated failures is based on the Beta-Binomial distribution. The Beta-Binomial distribution has been used to model other correlated events [Griffiths1973][Nicola1990]. The distribution is similar to the model based on conditional probabilities in the sense that it uses average availability and a correlation level as parameters. The Beta-Binomial distribution is computed by randomizing the parameter (*p*=failure probability) of the Binomial distribution. In general the randomized Binomial distribution is,

$$b_N(i) = \int_0^1 \binom{N}{i} p^i (1-p)^{N-i} f_p(p)\,dp$$

The intensity function $f_p(p)$ gives the probability distribution that a fraction *p* of all nodes fail. It is a unit impulse at Average (*p*) when there is no correlation. For the Beta-Binomial distribution, the intensity function is,

$$f_p(p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}, \qquad 0 < p < 1, \qquad \alpha,\beta > 0$$

where $B(\alpha,\beta)$ is the beta function,

$$B(\alpha,\beta) = \int_0^1 p^{\alpha-1} \quad (1-p)^{\beta-1} \quad dp$$

Finally, the formula of the probability that *i* out of *N* machines fail is

$$b_N(i) = \binom{N}{i} \frac{[(\pi+(i-1)\vartheta)(\pi+(i-2)\vartheta)\ldots\pi][(\chi+(N-i-1)\vartheta)(\chi+(N-i-2)\vartheta)\ldots\chi]}{(1+(N-1)\vartheta)(1+(N-2)\vartheta)\ldots1}$$

$$where \quad i = 0,1,2,...N$$

$$\pi = (1-\chi) = (1-Avg.Avail) = \frac{\alpha}{\alpha+\beta} = \quad Average(p)$$

$$\vartheta = Correlation \quad Level = \quad \frac{1}{\alpha+\beta}$$

In the literature there are different methods for estimating the correlation level from the data [Griffiths1973]. In this context, the correlation level is used as a measure of variation in p. To compute the correlation level, we adopted the method where first the availability measurements are used to determine the value of $b_2(1) + b_2(2)$ empirically (i.e., the case of n=2, m=2). The measured value of $b_2(1) + b_2(2)$ is used to solve for the unknown correlation level ϑ. Once the correlation level is determined the distribution can be used to estimate the availability of any scheme (n, m). As an example of using the Beta-Binomial distribution, the availability of the scheme with n=4 and m=2 would be (1- $b_4(3)$-$b_4(4)$), which means that it can survive at most 2 failures.

The Beta-Binomial distribution reduces to the Binomial distribution (independent failures) when the correlation level is zero. In that sense, the correlation level is different from the correlation level used in the model based on conditional probabilities (i.e., in that model independent failures means Correlation Level = (1-Avg.Avail.)).

4.4 Comparison of the Models

In this section, we compare the two models we described and the classic model.

Comparison Using the Web Servers Datasets

Figures 7, 8, 9 and 10 show the comparison for the Web Servers Dataset-1, Dataset-2, Dataset-3 and Dataset-4, respectively. The graphs have 4 types of bars and show the comparison of the actual availability of each scheme, and the estimations of the model based on conditional probabilities, the Beta-Binomial distribution and the classic model. The figures show that the model based on conditional probabilities and the Beta-Binomial distribution are, in general, more accurate than the classic model. The Beta-Binomial distribution and the model based on conditional probabilities are comparable. On average, the model based on conditional probabilities is more precise.

The average and maximum absolute errors (units are nines of availability) for the Web Servers datasets are listed in Table 3. For all four datasets, the model based on conditional probabilities outperformed the other two models. A point to note is that, although the maximum error of the model based on conditional probabilities for dataset 4 is ~2 nines, this error occurs for the scheme with n=10 m=1. The actual availability of this scheme is ~11 nines (0.00031536 seconds unavailability per year) the estimated availability is ~13 nines (0.0000031536 seconds unavailability per year). Therefore, examining in which scheme the error has occurred is more informative.

	MODEL BASED ON COND. PROBABILITIES		BETA-BINOMIAL		CLASSIC MODEL	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Dataset-1	0.127	0.297	0.229	0.717	2.951	11.407
Dataset-2	0.32	1.28	0.529	2.25	2.671	11.001
Dataset-3	0.32	1.202	0.942	3.781	1.406	5.940
Dataset-4	0.32	1.961	0.711	3.777	0.56	3.124

Table 3. Absolute Errors for the Web Servers (Unit: nines of availability).

We now analyze the error of the model based on conditional probabilities in detail. Figure 11 shows only the schemes with *m*=1 for Dataset-1. The error for the scheme *n*=2 *m*=1 is due to the fact that the model uses only averages and does not consider the variance of the availabilities and the variance of the correlations of the servers. The following example illustrates the problem. Assume there are only 2 servers in the system, the average availability is 0.95 and the correlation level is 0.5. The model's estimation of the availability for the scheme *n*=2 *m*=1 is:

= 1 - P (both servers are down)

= 1 - 0.05 * 0.5

= 0.975

Further assume that, in reality, one of the servers has an availability of 0.975 and the other has an availability of 0.925 (the average is 0.95). In this case, assuming the correlation level is 0.5, the two conditional probabilities can be calculated as follows (i.e. Corr_1 = P (A is down | B is down) and Corr_2 = P (B is down | A is down)):

0.025 * Corr_1 = 0.075 * Corr_2   (the portion of the downtimes that overlap should be equal)

(Corr_1 + Corr_2)/2 = 0.5   (0.5 = Correlation level)

Therefore, Corr_1 = 0.75, Corr_2 = 0.25 and the actual availability is;

= 1 – 0.025 * Corr_1   (which is equal to 1- 0.075 *Corr_2)

= 0.98125

which is higher than the model's prediction.

A more detailed discussion of this issue can be found in the next section. For the schemes with *m*=1, if *n* is low, the model under-estimates, but it over-estimates for higher *n*'s. This increase is because the model under-estimates the higher x-way failure probabilities.

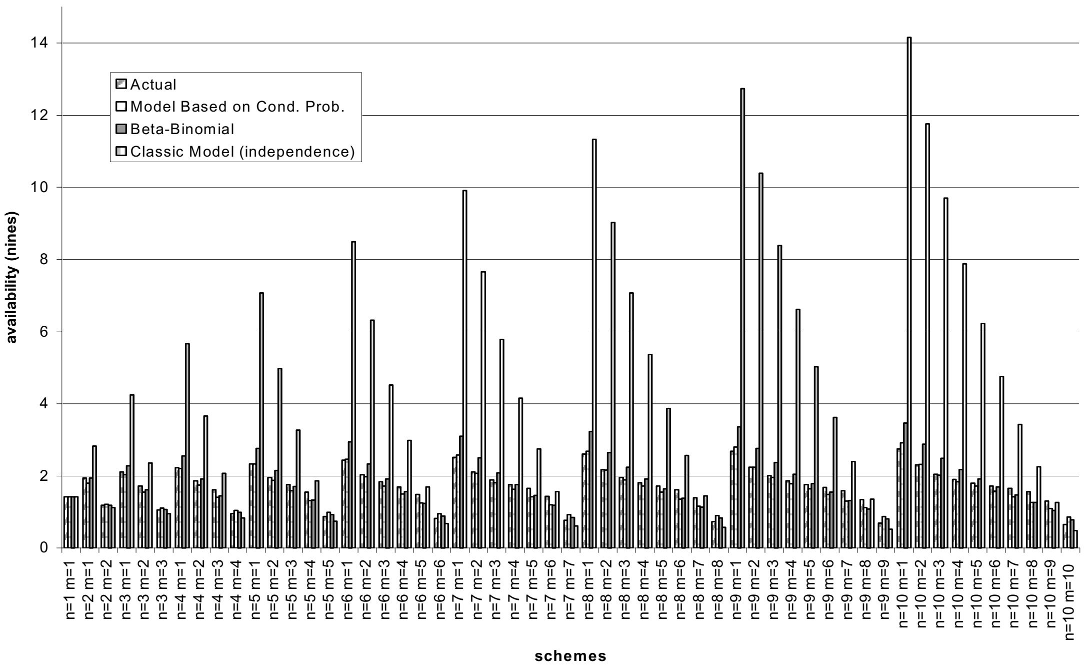

Figure 7. Comparison of the 3 Models using Web Servers Dataset-1

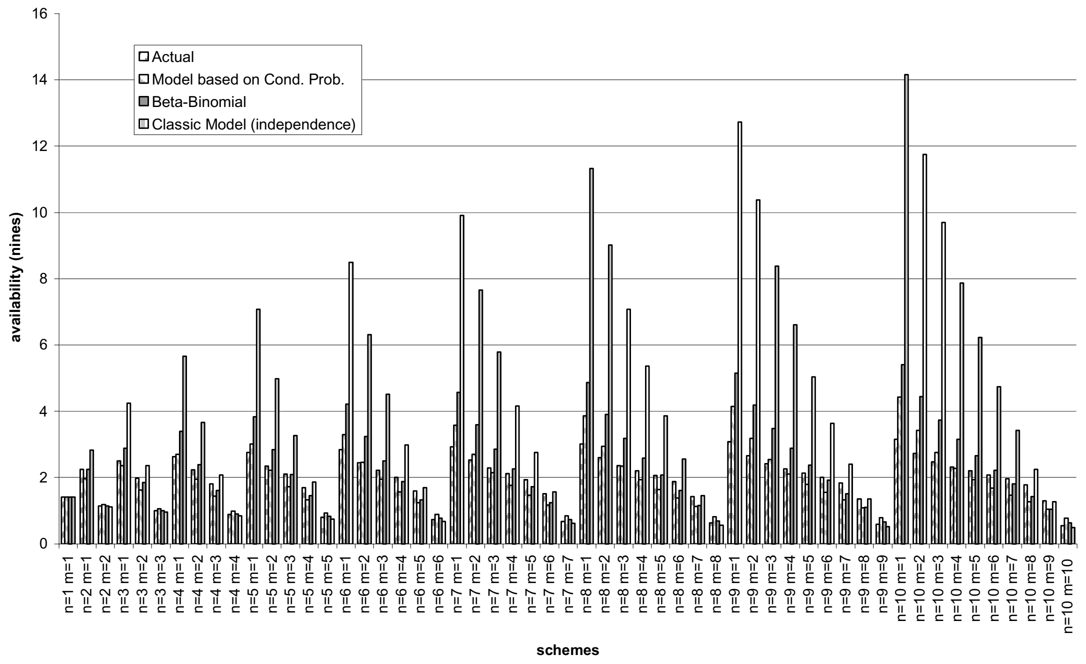

Figure 8. Comparison of the 3 Models using Web Servers Dataset-2

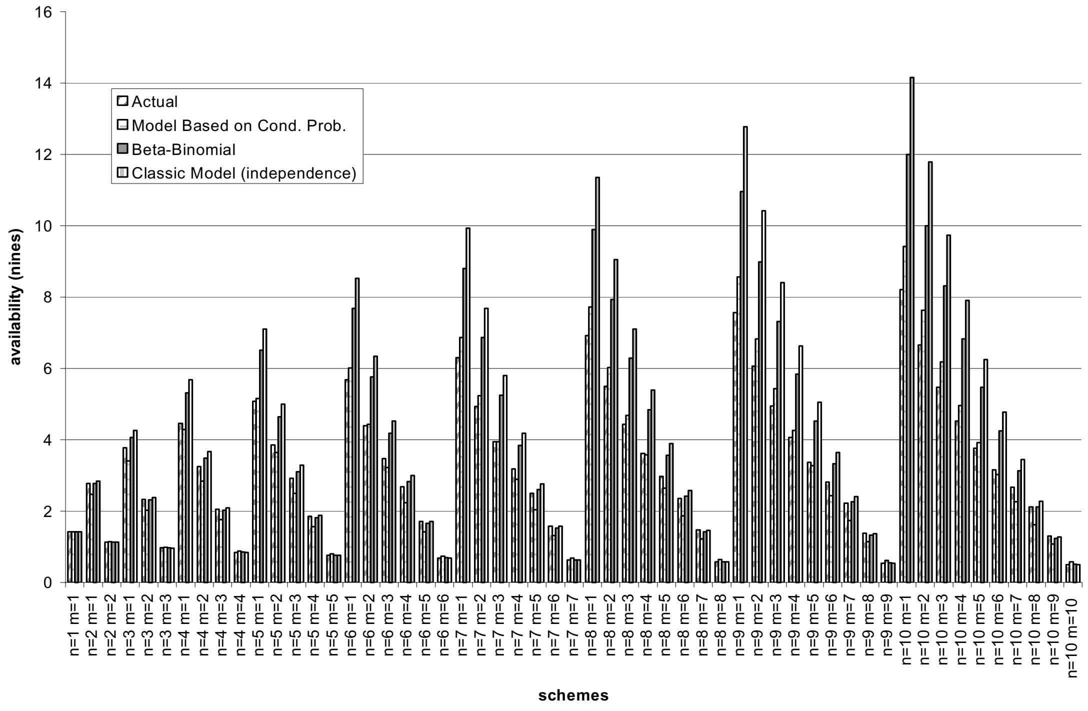

Figure 9. Comparison of the 3 Models using Web Servers Dataset-3

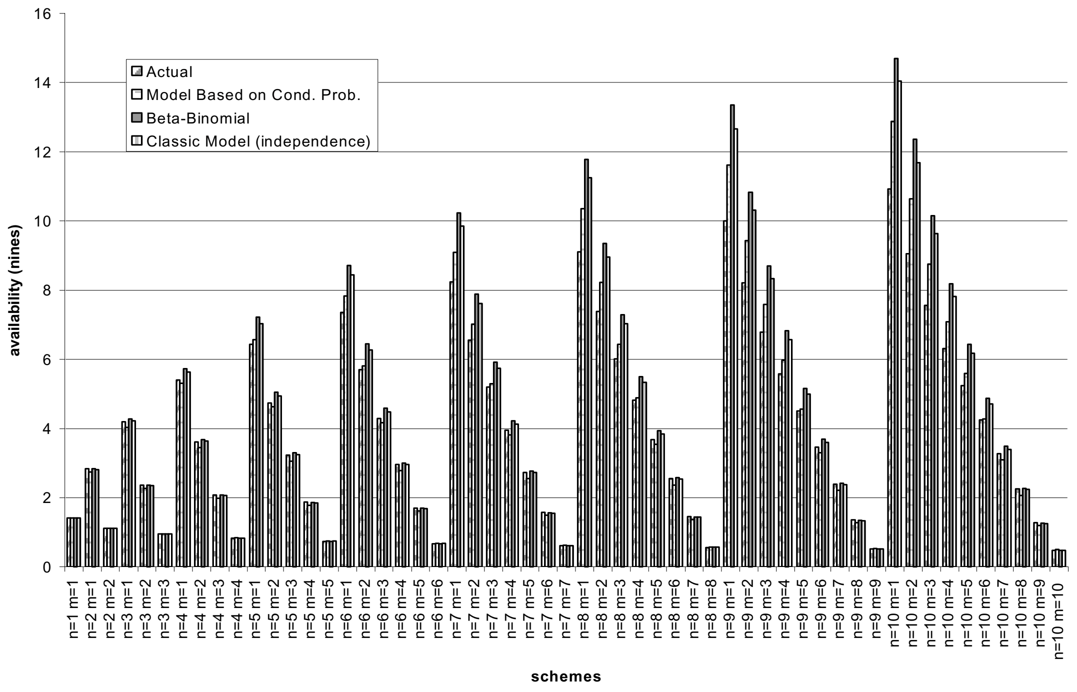

Figure 10. Comparison of the 3 Models using Web Servers Dataset-4

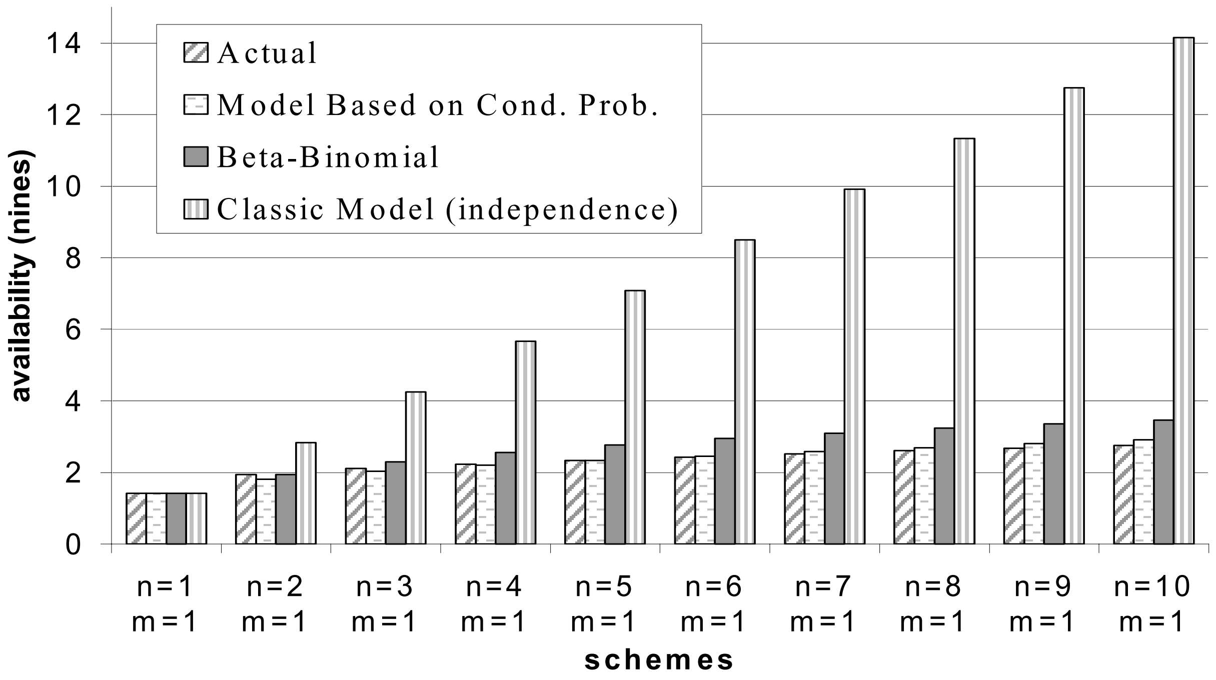

Figure 11. Availabilities of Schemes with m=1 for the Web Servers Dataset-1

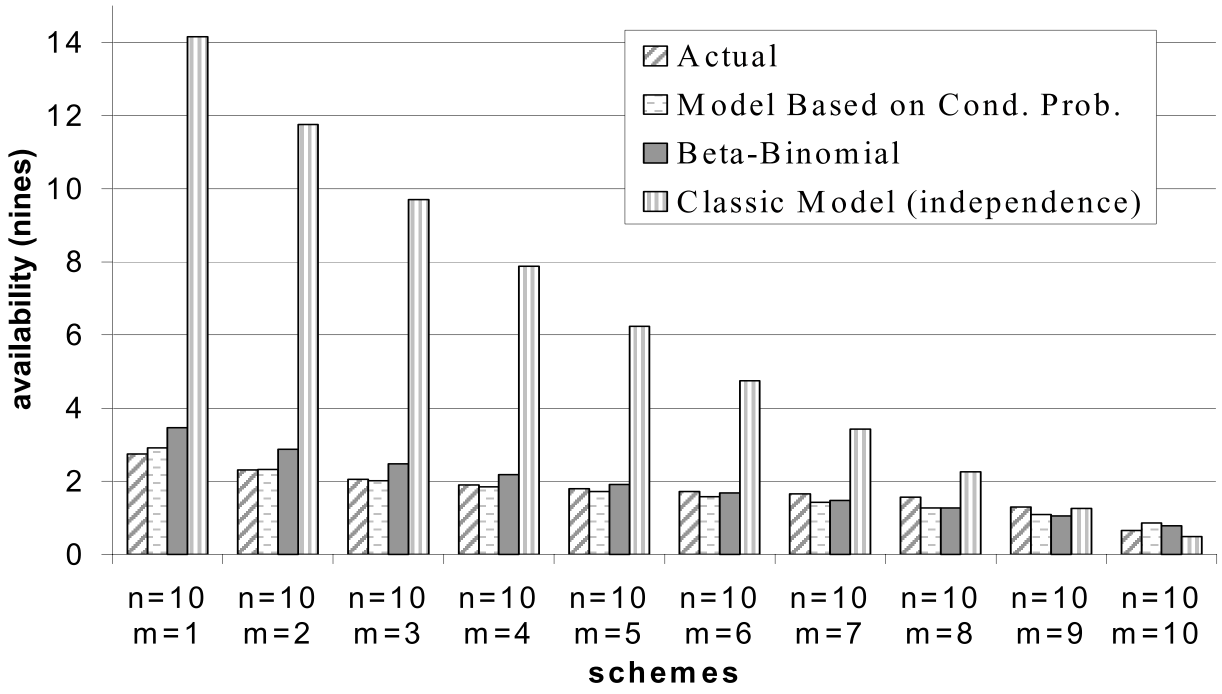

Figure 12. Availabilities of Schemes with n=10 for the Web Servers Dataset-1

In Figure 12, the schemes with *n*=10 for Dataset-1 are shown. For *m* between 1 and *n*-1, the model first over-estimates and then underestimates. Note that the Beta-Binomial distribution exhibits the same behavior. This is mainly due to two factors. First, note that a decrease in failure correlation increases the availability of schemes with large (*n*–*m*)'s more than schemes with small (*n*–*m*)'s. Since the model under-estimates the correlation, availability of schemes with large (*n*–*m*)'s tend to be over-estimated. Second, the model assumes that all the nodes are identical and assumes smooth transitions from one x-way correlation to another. In reality, some sets of servers are more correlated to each other than to others; thus, using average correlation and average availability leads to inaccuracies.

For schemes with *n*=*m*, the two proposed models always over-estimate. This is because the models consider only the average availability. In reality the schemes with *n*=*m* are bound by the server with minimum availability. In summary, a model that uses only average availability is bound to over-estimate the availability of schemes with *n*=*m* unless it significantly under-estimates the higher x-way correlations (e.g., independence).

Comparison Using the Desktops Datasets

Figures 13 and 14 show the comparison of the 4 models using the Desktops Dataset-1 and Dataset-3. The failures of the machines in this set were nearly independent. The importance of using the Desktops measurements in our comparison is to show that the proposed models work for datasets that are nearly independent as well as highly correlated datasets. Both of the models were accurate for this case. The interesting point here is that the Beta-Binomial distribution has different behaviors for the two datasets. The way Beta-Binomial determines correlation is by looking at the actual availability of the scheme *n*=2 *m*=2. Therefore, its accuracy is limited by the accuracy of the correlation calculated. Table 4 shows the errors of the 3 models for the four datasets.

	MODEL BASED ON COND. PROBABILITIES		BETA-BINOMIAL		CLASSIC MODEL	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Dataset-1	0.018	0.148	0.023	0.238	0.086	0.494
Dataset-2	0.023	0.188	0.021	0.205	0.087	0.489
Dataset-3	0.032	0.263	0.001	0.017	0.076	0.456
Dataset-4	0.033	0.259	0.001	0.022	0.072	0.427

Table 4. Absolute Errors for the Desktops (Unit: nines of availability).

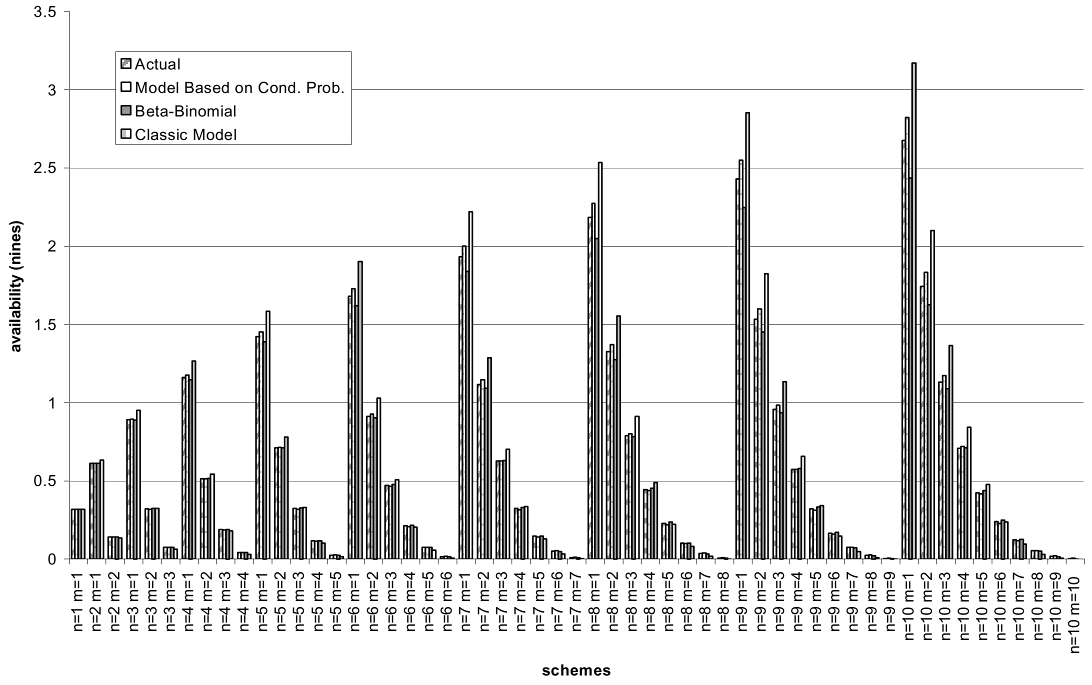

Figure 13. Comparison of the 3 Models using Desktops Dataset-1

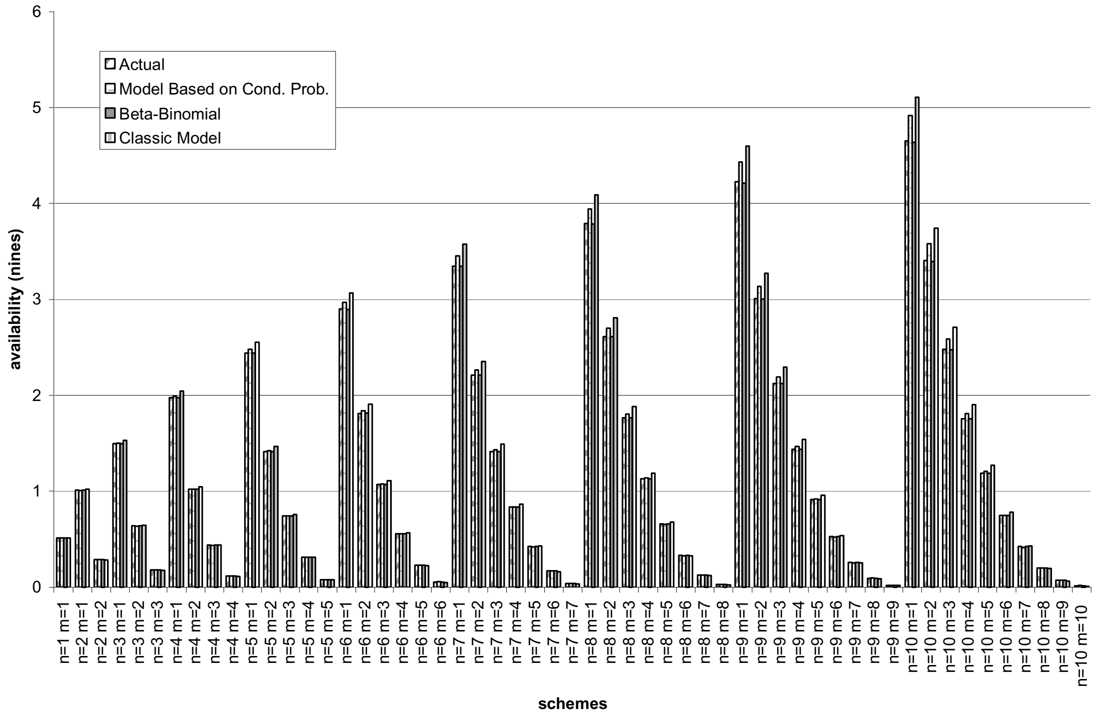

Figure 14. Comparison of the 3 Models using Desktops Dataset-3

4.4.1 Sensitivity of the Model Based on Conditional Probabilities

In section 4.2, we gave an example that showed that the classic model was unable to cover the range of possible availabilities for the case of two machines with availability 0.99. This was due to the independence assumption. Since the model based on conditional probabilities considers correlation, it is able to account for the full range (see Table 5).

Correlation Factor	New Model's estimation
perfectly correlated 1	0.99
independent 0.01	0.9999
inversely correlated 0	1.0

Table 5. Effect of having a Correlation Factor

However, there are cases when the model is not perfectly accurate. We now analyze the new model in depth.

Using Averages

The model uses only one parameter that defines the availability of the nodes: the average availability. It does not consider the variance among the servers' availabilities. A heuristic such as, using the geometric mean instead of the arithmetic mean could be used. The geometric mean is always less than or equal to the arithmetic mean. However, the geometric mean is not any more general then the arithmetic mean. Depending on whether in the dataset the machines with lower availabilities have higher correlation or the machines with higher availabilities have higher correlation either the geometric mean or the arithmetic mean could result in more accurate estimations. Also the accuracies of the two methods depend on what the scheme parameters are. Therefore, no matter how the availability parameter is calculated, the model is agnostic to the variance among the servers and this has an effect on the accuracy of the model's estimations. But note that we do not want to have an extra variance parameter in the model since this would complicate the model.

To support our hypothesis that not considering variances lead to inaccuracies, we conducted a second experiment using the Web Server measurements. In this experiment we used a subset of the servers (14 of them) that have similar availabilities (i.e., low variance). Figure 15 shows the graph obtained. In this case the model's estimations are much closer to the actual availabilities. The average error is 0.053 nines and the maximum error is 0.117 nines.

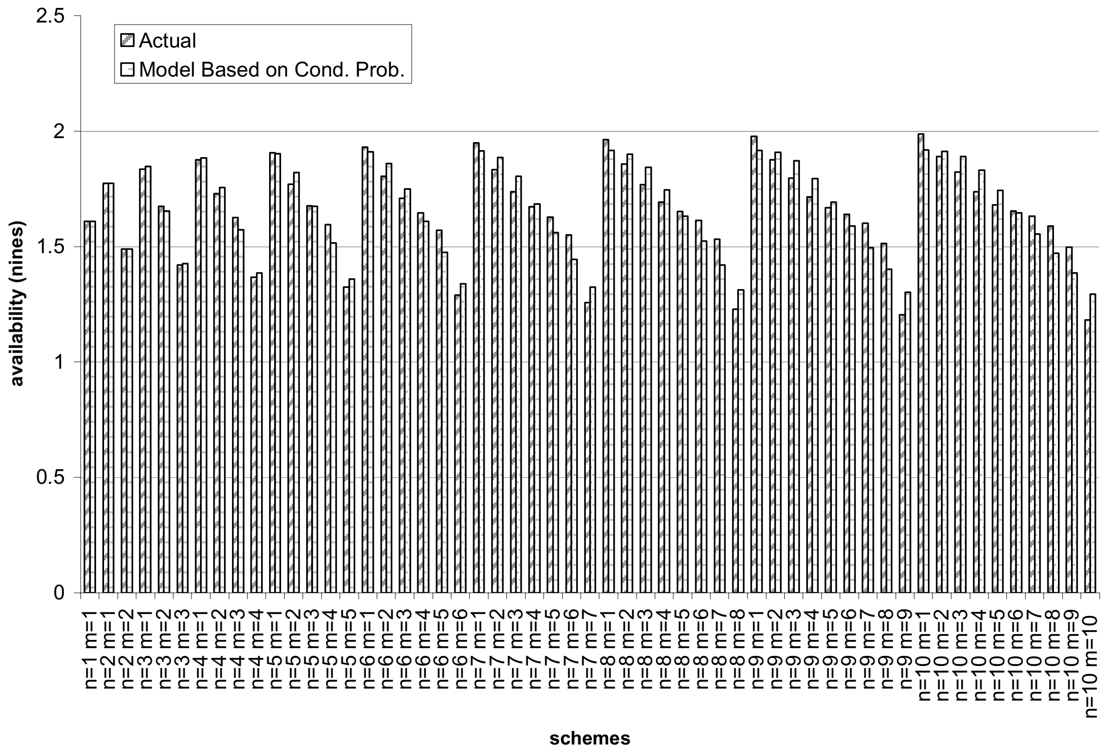

Figure 15. Comparison Using a Subset of the Web Servers

Incomplete Correlation Information

As mentioned earlier, the model estimates the higher-level correlations using the two parameters and this estimation can lead to inaccuracies. Our results in Section 4.4, show that the model based on conditional probabilities is accurate for the Web Servers and Desktops datasets. For datasets 3 and 4 from the Web Servers measurements, although the 2-way correlation (i.e. the correlation level) is not representative of the higher x-way correlations, the model's estimations are reasonably accurate. It is conceivable that some cases may have zero 2-way correlation but high x-way correlations. In such a case since the correlation level indicates no correlation the accuracy of the model will be low.

4.5 Related Availability Modeling Work

In the literature, there are several approaches to modeling the availability of a single machine. Most of these studies are based on Markov Chain models [Garg1999][Kalyanakrishnam1999]. For example, in [Kalyanakrishnam1999], each state represents a level of functionality of the machine, such as "reboot", "connectivity problems", "adapter problems", "disk problems", "shutdown" etc. and the weight of transitions between states is determined empirically.

There are similar state-based modeling studies for multiple machines. Those techniques are useful for cases where there are a small number of machines and it is possible to list all the states of the entire system. In [Kalyanakrishnam1999], they model mail servers using a finite state model. Specifically they have a primary machine and a number of backup machines. They identify all the possible states and the transition probabilities (empirical values) between the states. They account for correlated failures by having a state that represents the case when more than one machine is unavailable. Their model is completely empirical. Such an approach is impractical for cases where there are hundreds of servers since it would be impossible to identify all of the states and the relationships between states.

In [Tang1992], they present an approach to correlation modeling that is similar to the model based on conditional probabilities. They also use conditional probabilities as a measure of correlation among failures of machines. They only model 2-way correlations, whereas in our case, to determine the availabilities of schemes with n>2, in addition to 2-way correlations higher-way correlations are also modeled. Therefore, their model does not involve or require any estimation of unknown probabilities.

In [Douceur2001c], effective system availability (ESA) is used to compare the availabilities of different schemes. The availabilities of individual files are calculated using the independence assumption. ESA is a weighted average of the availabilities of the files in the system. Files with low availabilities have a higher weight. For N files, each with availability a_i (number of nines), ESA is calculated as follows,

$$ESA = \ -\log\frac{1}{N} \ \sum_{i=0}^{N-1} 10^{-a_i}$$

They also calculate the actual system availability and they compensate for the difference between the actual ESA and the calculated ESA by finding the following relationship [Douceur2001c].

$$ESA_{actual} = -0.078\ (ESA)^2 + 1.5\ ESA - 0.84$$

In [Douceur2001c], they do not claim that their model is generic, and they mention that this model is valid for ESA values in the range of 2 to 6 nines. When we try their model for our Web Servers dataset, we get errors such as negative numbers and erroneous mappings, such as ESA~14 becomes ~4.8 but ESA~9 becomes ~6.3. Because of such errors and the inaccuracy of their model for our Web Servers dataset, we do not include their model in the comparison.

In [Littlewood1989], they model failures in multi-version software. Their approach results in a complicated model that provides little guidance to system designers.

5 Ordering of the Schemes

In Section 3, we described how scheme selection is done. In scheme selection, an important point is the ordering of the schemes. The reason why the ordering of the schemes is important is due to the fact that the user could have estimated the average availability of her system incorrectly. Thus, if this effects the ordering of the schemes, the scheme the user selects (e.g., the 50 percentile scheme) using the incorrect estimate would not be the optimal scheme.

5.1 Analysis

In this section, we investigate how average availability and the correlation level affect the ordering of the schemes. By "ordering", we mean the schemes sorted according to their availabilities. In our analysis, we use the model based on conditional probabilities and we only consider schemes with $n \leq 10$.

Varying Average Storage Node Availability

Here we consider four cases: varying average storage node availability from 0.90 to 0.95, from 0.90 to 0.99, from 0.90 to 0.999 and from 0.90 to 0.9999. The failures are assumed to be independent. We classify schemes according to their change in rank (i.e., if a scheme is 3rd and becomes 5th, the change in its rank is 2). Table 6 shows the number of schemes in each class. The first row shows the number of schemes that have a rank change of 0 (i.e., same rank). The table shows that overall the rank changes are not significant; only a few schemes change significantly. Also note that the ordering stabilizes after 0.999 (3 nines).

	Number of Schemes (In total 55 schemes)			
Change in Rank	**From 0.90 to 0.95**	**From 0.90 to 0.99**	**From 0.90 to 0.999**	**From 0.90 to 0.9999**
0	29	16	16	16
1	18	18	14	14
2	8	12	13	13
3	0	6	6	6
4	0	1	2	2
5	0	2	2	2
6	0	0	2	2

Table 6. Changes that result from varying the availability level

Now, we analyze the significance of changes in ranks. Assume the ordering of the schemes using the correct value for average node availability is A, and the ordering of the schemes using an incorrect value for average node availability is B. In ordering B, the scheme at rank x (scheme B) may be different from the scheme at rank x (scheme A), in ordering A. The absolute difference (D) in the availabilities of the two schemes, in the correct setup, defines the significance of the change in ranks. The difference between the two orderings A and B is the average of all D's over all x (x=1, 2 … 55). For the four cases, defined above, we assume 0.90 is the incorrect average node availability. The statistics, for the four cases, are listed in Table 7. As the difference between the correct and the incorrect average node availability increases, the significance of the difference increases.

Ordering B (Incorrect Average Storage Node Availability)	**Ordering A (Correct Average Storage Node Availability)**	**Average (D) (nines)**	**Maximum (D) (nines)**
0.90	0.95	0.08	0.36
0.90	0.99	0.39	1.31
0.90	0.999	0.81	2.30
0.90	0.9999	1.24	3.30

Table 7. Differences Between Orderings

To discover trends in the ordering of schemes with respect to average storage node availability we performed another experiment. Figures 16 shows how the availability for each scheme changes as average availability increases. For clarity, only a carefully chosen subset of the schemes is shown. Note that there are patterns in the change of ranks. For low availability levels, schemes with (n_1, m_1) where m_1=1 are better than schemes with (n_2, m_2) where $n2=n1+x+y$, $m2=m1+x$ (x>0 and y>0). But as availability increases, schemes with (n_2, m_2) start outperforming schemes with (n_1, m_1). For example, the scheme with n=4 m=1 eventually is surpassed by schemes n=10 m=4, n=8 m=3, n=9 m=4, n=10 m=5, n=6 m=2, n=7 m=3, n=8 m=4, n=9 m=5, n=10 m=6. The common property among these schemes is that the difference (n-m) for all of these schemes is greater than that for the scheme with n=4 m=1, where (n-m)=3. In general, schemes with a larger (n-m) eventually outperform schemes with a lower (n-m). For example, the scheme with n=10 m=4 eventually outperforms the schemes with n=5 m=1 and n=6 m=1. This observation also shows that treating (n/m) as the replication factor in terms of the availability is incorrect.

Also, among the schemes that have the same (n-m); the ones that have lower n's are the winners. That is, n=4 m=1 outperforms n=5 m=2. Another observation is that, after 3 nines of average storage node availability, the schemes that have the same (n-m) form a cluster. In other words, the schemes converge to a total ordering. Within a cluster the ones with lower n's have higher availabilities. The final observation here is that, although there are rank changes between two average storage node availabilities, the absolute scheme availability changes are not large. However, large changes in average storage node availability may result in significant differences.

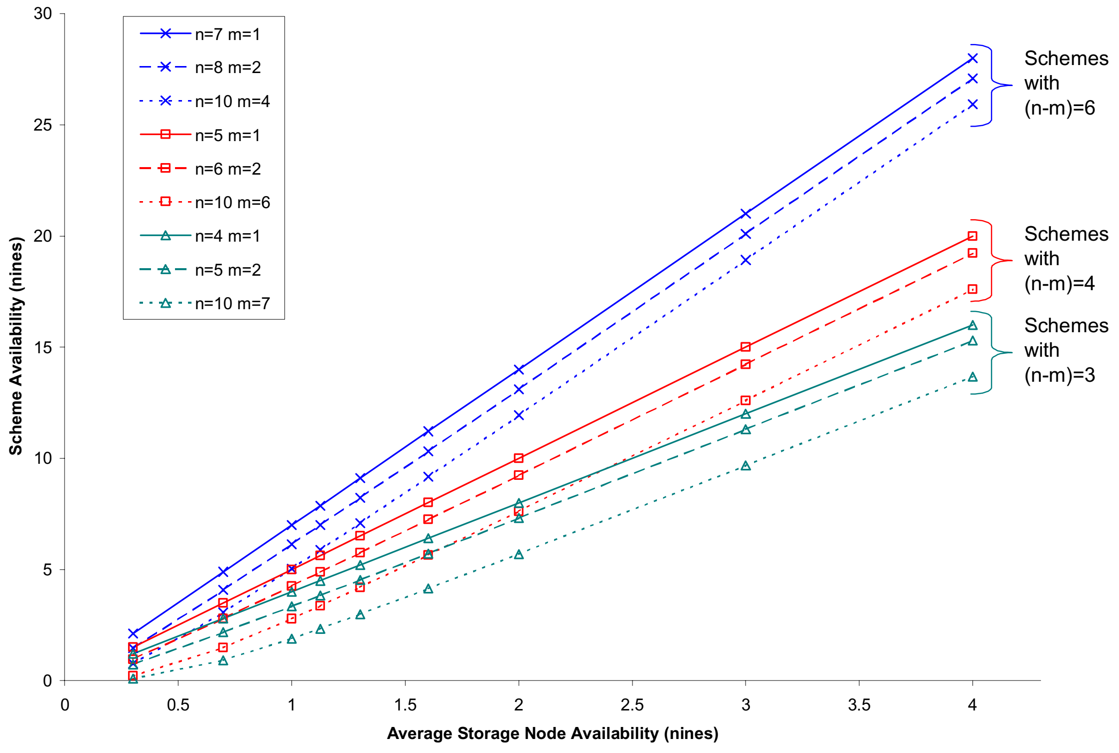

Figure 16. Scheme Availabilities as Average Storage Node Availability changes

Varying the Correlation Level

Next, we examined the effect of the correlation level on the ordering of the schemes. We considered two cases: the first case is when the correlation level changes from 0.05 to 0.1 and the second case is when it changes from 0.05 and 0.25. Storage node availability is 0.95 (1.3 nines). We classify schemes according to their change in rank (i.e., if a scheme is 3rd and becomes 5th, the change in its rank is 2). Table 8 shows the number of schemes in each class. The first row shows the number of schemes that have rank change 0 (i.e., same rank). Compared to the previous case, there are more changes in ranks. This is mainly because when there is correlation most schemes have a lower availability, and the availabilities of different schemes are closer to one another.

To analyze the significance of the changes in ranks we performed an experiment similar to the one we did for the previous case. The statistics are shown in Table 9. Note that, the differences here are less than the previous case. But, since the actual availabilities (Figure17) in this case are less than the previous case, the differences here are more important than the previous case.

	Number of Schemes (In total 55 schemes)	
Change in Rank	**From 0.05 to 0.1**	**From 0.05 to 0.25**
0	18	11
1	24	12
2	10	7
3	2	10
4	1	7
5	0	3
6	0	2
7	0	1
8	0	2

Table 8. Changes that result from varying the correlation level

Ordering B (Incorrect Correlation Level)	Ordering A (Correct Correlation Level)	Average (D) (nines)	Maximum (D) (nines)
0.05	0.1	0.10	0.42
0.05	0.25	0.17	0.56

Table 9. Differences Between Orderings

To discover trends in the rank changes, we examined individual scheme availabilities as the correlation level changes from 0.05 (being independent) to 0.50. Figure 17 shows the corresponding graph. For clarity purposes, the graph shows only the availabilities of schemes with (*n*-*m*) equal to 1, 3 and 6. The observations drawn are,

- The schemes that have the same (*n*-*m*) are always ordered according to their *m* values (the scheme with *m*=1 being the best).
- The schemes with larger (*n*-*m*) are influenced significantly by correlation. Also not shown in the graph, the availabilities of schemes with *n*=*m* increase as the correlation level increases.
- As correlation approaches 1, the availabilities of all of the schemes converge to the average storage node availability (1.3 nines in this case).
- The schemes with low (*n*-*m*)'s and low *m*'s outperform schemes with large (*n*-*m*)'s and large *m*'s. For example, the scheme with *n*=2 *m*=1 passes the scheme with *n*=10 *m*=7 (Also, not shown in the graph, it passes the schemes with *n*=7 *m*=4, *n*=10 *m*=6, *n*=10 *m*=5 etc.). This is because schemes with large (*n*-*m*)'s, are affected more from correlation.

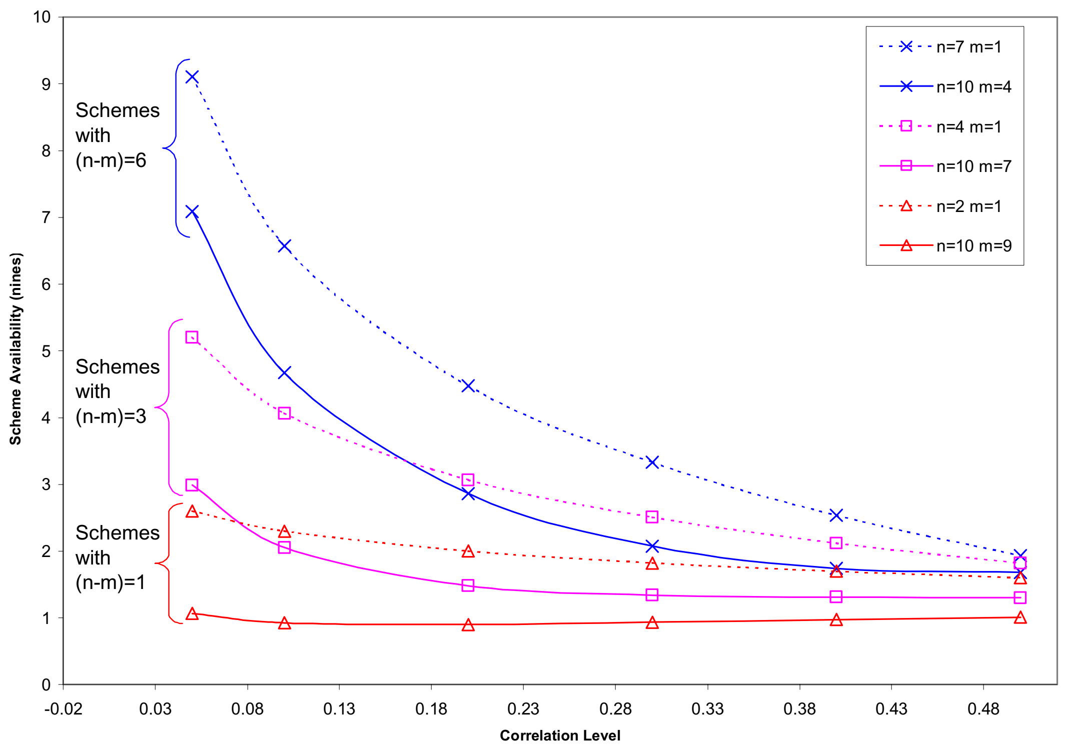

Figure 17. Scheme Availabilities as Correlation Level changes

6 Placement of Related Files

In designing a distributed system, the ways in which files (shares) are stored may affect the availability of the files. Previous work in databases analyzes how to store the stripes of the primary and the backup copies. Striping is a scheme with parameters *m*=*n* and *p*=1 (e.g., *n*=3 *m*=3 *p*=1). The two well-known placement methods are chained de-clustering and interleaved de-clustering [DeWitt1990]. The idea in these two methods is to store the corresponding stripes of the primary and backup copies on different machines in a manner that in the face of failures all the data will still be available and the load will be balanced among the remaining nodes. In their case, all of the machines are identical and they only have to store the primary and backup copies of the database. In [Douceur2001b], they use a set of desktop machines with different availabilities and their measurements indicate that the failures between machine failures are nearly independent. Since an exhaustive search of the best placement of files is not feasible, they compare three different algorithms, which achieve near optimal placement solutions.

An interesting problem with file placement that is overlooked in the literature is the placement of files that are related. Two files are related if one file can be accessed only if the other file is available. In systems that have a directory structure, a file can only be accessed by traversing the corresponding directory path. In PASIS [Wylie2000], directories are stored the same way files are stored (i.e. using threshold schemes). In order to get to a file, all of the directories on the path have to be available. To achieve optimal overall availability, the shares of the related files should be stored on the same set of storage nodes.

6.1 Analysis

Independent Failures

In this section, the placement of related files is analyzed in depth. First, we examine the case when failures are independent. Assume, there are N related files, the availabilities of the storage nodes in the system are equal to (Avail.) and the scheme used is n=1 m=1. One extreme case is to store all the files on the same storage node. In that case, the overall availability is equal to the availability of a single storage node. The other extreme case is to store each file on a separate node (as long as there are sufficient number of nodes in the system). The overall availability in that case is $(\text{Avail.})^N$. The difference in availability between the two cases is,

$$(\textit{nines}\ (\text{Avail.}) - \textit{nines}\ ((\text{Avail.})^N))$$

where *nines* is the function that converts the corresponding availability value to nines. For N=100 and Avail.=0.95, the difference is equal to 1.298 nines. Now, let us look at different schemes.

In Figure 18, we assume that average storage node availability is 0.95 and the storage node failures are independent (we later relax this assumption). The x-axis shows the number of related files stored and the y-axis shows the difference in availability between the two extreme placement strategies. The first strategy is to store the shares of k files on the same set of n storage nodes. The second strategy is to store the shares of k files on completely different sets of nodes (i.e. on a total of $n*k$ storage nodes). The former always offers better availability and the graph shows the difference between the two strategies as the number of related files (k) increases. For the scheme with n=1 m=1, the difference approaches 1.3 nines, as the number of related files increases. For the scheme with n=10 m=1, the difference approaches 13 nines. This large difference is mainly due to the independence assumption. For the schemes with n=m, since the availability achieved is lower the line tends to flatten earlier. For the scheme with n=10 m=10, the maximum difference is ~0.4.

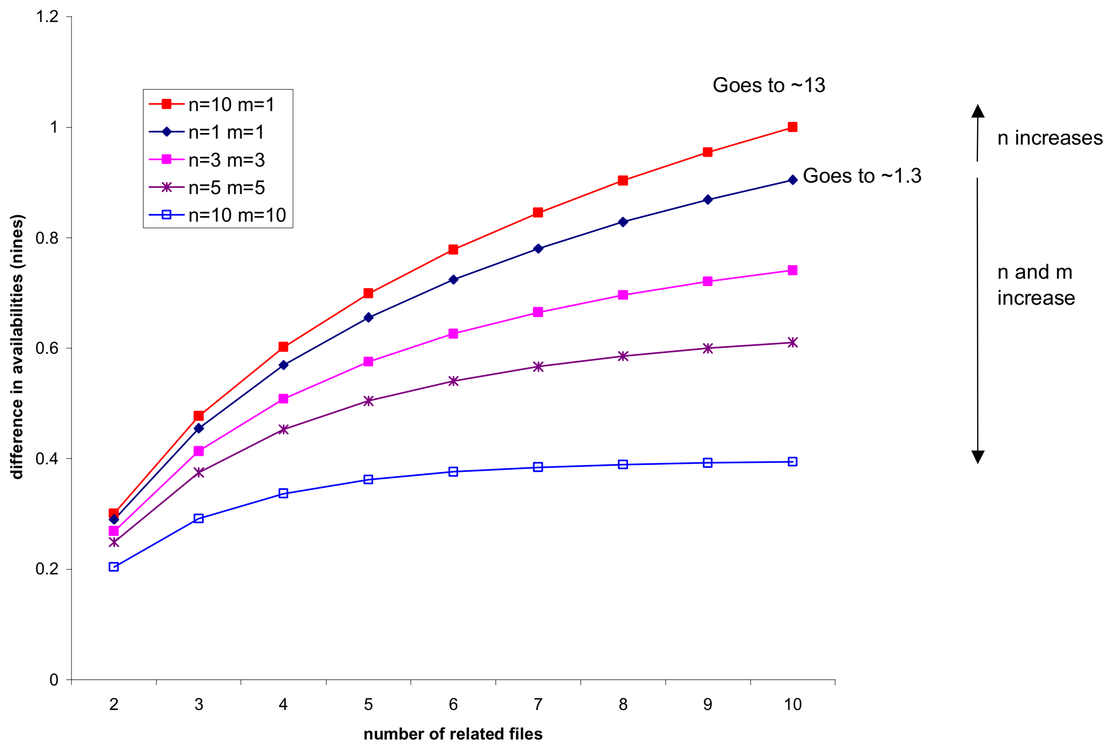

Figure 18. Effect of Having Related Files (Assuming Independent Failures)

Correlated Failures

Now, let us examine the same problem when the failures between storage nodes are correlated. First, using our model, we show how the availability is calculated when the shares of different files are stored on distinct sets of storage nodes.

Consider the scheme *n*=2 and *m*=1 and assume there are 2 files stored on a total of 4 storage nodes (Figure 19). The shares of one of the files are stored on Node 1 and 2. The shares of the other file are stored on Node 3 and 4. If 3 or more nodes that are available, then the 2 files are available. If there are only 2 nodes available nodes, then for the two paths, shown in the figure, the files are unavailable. Note that, this is different from just having 2 of the 4 nodes up. Using this method, we analyzed the related files issue for the case when the average storage node availability is 0.95 and the correlation level is 0.25. The corresponding graph is shown in Figure 20.

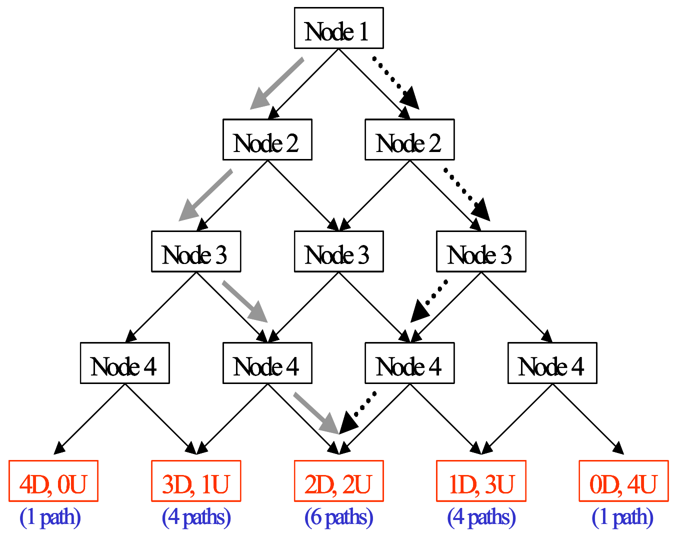

Figure 19. Two file example for the scheme with n=2 and m=1

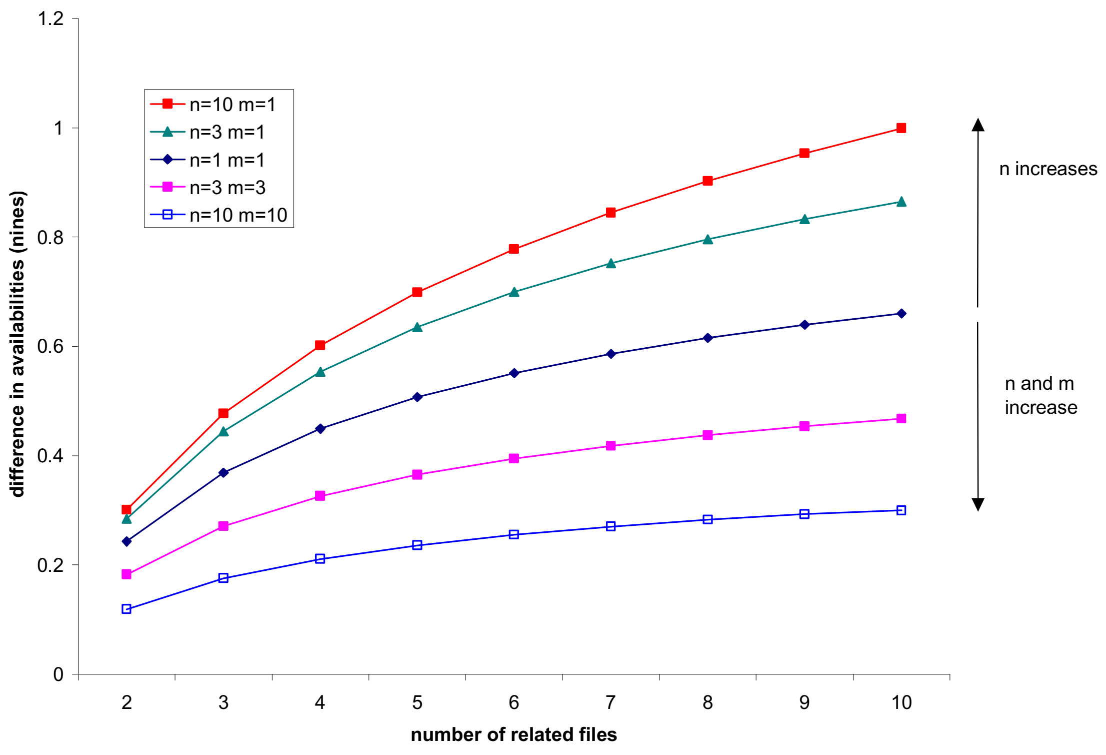

Figure 20. Effect of Having Related Files (Failures are Correlated)

The most prominent difference of Figure 20 from Figure 18 is that the difference (i.e., the value shown on the y-axis) for the schemes with $n=m$ is lower. The reason is, when correlation is introduced, nodes tend to fail simultaneously. Thus, when the files are stored on different sets of nodes they benefit from correlation. In the extreme case when correlation is 1, the difference is zero. For schemes with $m<n$, the difference is still important. Also, it is worthwhile to note that here the actual availabilities are lower. Therefore, a 1 nine difference in this figure is more crucial compared to Figure 18. For example, consider the scheme with $n=10$ $m=1$, for the case when there are 10 related files. When the failures are independent the difference is (13.01-12.01), but, when the correlation is 0.25, the difference is (4.89-3.89). In the former, the difference accounts for ~0.0000283824 seconds unavailability per year, whereas in the latter the difference is ~1.02 hours unavailability per year. Based on this analysis, it is more important to take care with regard to related files when the failures are correlated.

This section shows that in doing file placement, the system designer should consider relations between files. The importance becomes more substantial when the number of related files increases.

7 Conclusions

In this report, we investigate availability modeling for data distribution schemes. We explain the weakness of the classic availability models, and present two models that overcome the inaccuracy caused by the independent failures assumption. Our key contribution is a pair of models that preserve the simplicity of the classic model and at the same time produce more accurate results. We show the efficacy of the model based on conditional probabilities as a design decision tool. We analyze the effects of availability and correlation on the ordering of the schemes. Finally, we investigate the problem of optimal file placement.

8 References

[Amir1996] Y. Amir, A. Wool "Evaluating quorum systems over the Internet", Proceedings of the Twenty-Sixth Annual International Symposium on Fault-Tolerant Computing, 1996

[Amir1998] Y. Amir, A. Wool "Optimal Availability Quorum Systems: Theory and Practice", Information Processing Letters, 1998

[Blakley1985] G. R. Blakley, C. Meadows "Security of Ramp Schemes", Advances in Cryptology – CRYPTO, pages 242-268, Springer-Verlag 1985

[Bolosky2000] W. J. Bolosky, J. R. Douceur, D. Ely, M. Theimer "Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs", Proceedings of the international conference on Measurement and modeling of computer systems, pp. 34-43, 2000

[DeWitt1990] S. Ghandeharizadeh, D. Schneider, H. Hsiao, A. Bricker, R. Rasmussen "The GAMMA Database Machine Project", IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 1, March 1990

[Douceur2001a] J. R. Douceur, R. P. Wattenhofer "Modeling Replica Placement in a Distributed File System: Narrowing the Gap between Analysis and Simulation", Proceedings of 9th ESA,, pp. 356-367, 2001

[Douceur2001b] J. R. Douceur, R. P. Wattenhofer “Competitive Hill-Climbing Strategies for Replica Placement in a Distributed File System”, Proceedings of 15th DISC, pp. 48-62, 2001

[Douceur2001c] J. R. Douceur, R. P. Wattenhofer “Optimizing File Availability in a Secure Serverless Distributed File System“, Proceedings of 20th IEEE SRDS, pp. 4-13, 2001

[Garg1999] S. Garg,, Huang, Y., Kintala, C. M. R. Kintala, K. S. Trivedi, S. Yajnik. “Performance and Reliability Evaluation of Passive Replication Schemes in Application Level Fault Tolerance”, Proc. Twenty-Ninth International Symposium on Fault-Tolerant Computing (FTCS-29), June 1999

[Goldberg1998] A. V. Goldberg, P. N. Yianilos “Towards an Archival Intermemory“, Proceedings of IEEE Advances in Digital Libraries, ADL 98

[Gray1991] J. Gray and D. Siewiorek. “High-Availability Computer System”, IEEE Computer, 24(9):39--48, September 1991

[Griffiths1973] D. A. Griffiths “Maximum Likelihood Estimator for the Beta-Binomial Distribution and an Application to the Household Distribution of the Total Number of Cases of a Disease”, Biometrics vol. 29, pp. 637-648, December 1973

[Kalyanakrishnam1999] M. Kalyanakrishnam, Z. Kalbarczyk, R. Iyer "Failure Data Analysis of LAN of Windows NT Based Computers", Proc. of 18th Symposium on Reliable and Distributed Systems, SRDS '99, Lausanne, Switzerland, pp.178-187, 1999

[Kubiatowicz2000] J.Kubiatowicz, D.Bindel, Y.Chen, S.Czerwinski, P.Eaten, D.Geels, R.Gummadi, S.Rhea, H.Weatherspoon, W.Weimer, C.Wells, B.Zhao “OceanStore: An architecture for Global-Scale Persistent Storage”, ASPLOS 2000

[Littlewood1989] B. Littlewood, D.R. Miller, “Conceptual modeling of coincident failures in multiversion software”, IEEE Transactions on Software Engineering, Volume: 15 Issue: 12, Pages: 1596-1614, Dec. 1989

[Luby1998] M. Luby, M.Mitzenmacher, M. Shokrollahi, D. Spielman, V. Stemann “Analysis of low density codes and improved designs using irregular graphs”, In Proc. Of ACM STOC, May 1998

[Malkhi2000] D. Malkhi, M. Reiter, A. Wool “The load and availability of Byzantine quorum systems”, SIAM J. Computing, 29(6):1889-1906, 2000

[Nicola1990] V. F. Nicola, A. Goyal “Modeling of Correlated Failures and Community Error Recovery in Multiversion Software”. IEEE Transaction on Software Engineering, Vol. 16 No.3, March 1990

[Plank1997] J. Plank “A tutorial on Reed-Solomon coding for fault-tolerance in raid-like systems”, Software Practice and Experience, 27(9):995-1012, September 1997

[Rabin1989] M. O. Rabin “Efficient Dispersal of Information for Security, Load Balancing and Fault Tolerance”, Journal of the ACM, 36(2):335-348 ACM April 1989

[Shamir1979] A. Shamir “How to share a secret”, Communications of ACM,22(11):612-613 ACM, November 1979

[Tang1992] D. Tang, R. K. Iyer “Analysis and Modeling of Correlated Failures in Multicomputer Systems”, IEEE Transactions on Computers, Vol. 41 No:5, May 1992

[Siewiorek1992] D. P. Siewiorek, R. S. Swarz “Reliable Computer Systems: design and evaluation”, Digital Press, Second Edition, 1992

[Wylie2000] J. J. Wylie, M. W. Bigrigg, J. D. Strunk, G. R. Ganger, H. Kiliccote, P. K. Khosla “Survivable information storage systems”, IEEE Computer, 33(8): 61-68, August 2000

[Wylie2001] J. J. Wylie, M. Bakkaloglu, V. Pandurangan, M. W. Bigrigg, S. Oguz, K. Tew, C. Williams, G. R. Ganger, P. K. Khosla ”Selecting the right data distribution scheme for a survivable storage system”, Technical Report CMU-CS-01-120 Carnegie Mellon University, May 2001

9 Appendix

List of the Web Servers used in the experiments:

www.cmu.edu	www.moscowtimes.ru	www.microsoft.com
www.cs.cmu.edu	www.ukindex.co.uk	www.ibm.com
www.ece.cmu.edu	www.ukdirectory.co.uk	www.panasas.com
www.berkeley.edu	www.mod.uk	www.sun.com
www.mit.edu	www.state.mn.us	www.hp.com
www.harvard.edu	www.state.hi.us	www.ananzi.co.za
www.cs.ucsd.edu	www.cao.go.jp	www.southafrica.co.za
www.utexas.edu	www.jda.go.jp	www.csu.edu.au
www.florida.edu	www.beijing.gov.cn	www.made-in-denmark.dk
www.nyu.edu	www.gio.gov.tw	www.snb.ch
www.gwu.edu	www.gov.za	www.ual.com
www.upr.clu.edu	pmindia.nic.in	www.aa.com
www.metu.edu.tr	www.indianembassy.org	www.wtca.org
www.encyclopedia.com	www.mfa.gov.tr	www.afghanistanfoundation.org
www.jeremyshaffer.com	www.turkey.org	www.onefootball.com
www.freebsd.org	armedforces.nic.in	www.unesco.org
www.linux.org	www.pakistan-embassy.com	www.afghan-web.com
www.gnu.org	www.sc.gov.jm	vlib.org
www.asiasource.org	www.cabinet.gov.jm	www.dictionary.com
www.newyorktimes.com	www.cuba.cu	www.bl.uk
www.canoe.ca	www.austrade.gov.au	www.literature.org
www.bbc.co.uk	www.admin.ch	www.druglibrary.org
www.japantimes.co.jp	www.gov.ru	www.oclc.org
www.nasda.go.jp	www.firstgov.gov	www.galaxy.com
www.sudan.net	www.whitehouse.gov	www.usenix.org
www.milliyet.com.tr	usembassy.state.gov	msdn.microsoft.com
www.pca.gov.sa	president.gov.al	www.acm.org
www.mg.co.za	www.gov.mk	www.online-library.org
www.afrika.no	www.presidency.gov.eg	www.library.cmu.edu
www.jpost.com	www.loc.gov	digital.library.upenn.edu
www.suntimes.co.za	www.nasa.gov	www.sba.gov
www.pressnet.or.jp	www.emc.com	www.amnesty.org
www.heraldsun.news.com.au	www.intel.com	www.prodigy.com