Designing for disasters

Kimberly Keeton, Cipriano Santos, Dirk Beyer, Jeffrey Chase, John Wilkes
Hewlett-Packard Laboratories, Palo Alto, CA and Duke University, Durham, NC

{kimberly.keeton, cipriano.santos, dirk.beyer, john.wilkes}@hp.com, chase@cs.duke.edu

Losing information when a storage device or data center fails can bring a company to its knees—or put it out of business altogether. Such catastrophic outcomes can readily be prevented with today's storage technology, albeit with some difficulty: the design space of solutions is surprisingly large, the configuration choices are myriad, and the alternatives interact in complicated ways. Thus, solutions are often over- or under-engineered, and administrators may not understand the degree of dependability they provide.

Our solution is a tool that automates the design of disaster-tolerant solutions. Driven by financial objectives and detailed models of the behaviors and costs of the most common solutions (tape backup, remote mirroring, site failover, and site reconstruction), it appropriately selects designs that meet its objectives under specified disaster scenarios. As a result, designing for disasters no longer needs to be a hit-or-miss affair.

1 Motivation

"The cascading blackout that swept through cities from Detroit and Cleveland to Ottawa and New York City was a harsh reminder to enterprises—to companies of all sizes, in fact—that disaster-prevention and business-continuity plans should be at the top of the priority list." – Techweb, August 22, 2003

Hardware breaks. Software has defects. Viruses propagate. Buildings catch fire. Power fails. People make mistakes. Although we prefer that these events never occur, it is only prudent to defend against them. The cost of data unavailability can be large: a quarter of the respondents to a 2001 survey [EagleRock2001] estimated their outage costs as more than $250 000/hour, and 8% estimated them as more than $1M/hour. The price of data loss is even higher. Recent high-profile disasters have raised awareness of the need to plan for recovery or continuity: since data is a critical asset, the key challenge is to construct *dependable* storage systems that protect data and the ability to access it.

Figure 1 illustrates the most commonly-used alternatives to protect and recover stored data. In all cases, a *primary copy* of the data is protected by making one or more *secondary copies*, which are isolated from failures that may affect the primary copy. Recovery involves either site *failover* to a secondary mirror, or data *reconstruction* at the primary site or a secondary site.

Each technique provides some of the necessary protection; combined, they create a large and complex design space. For example, a popular combination combines internally-redundant disk arrays (RAID), remote mirroring, snapshot and backup to tape: RAID protects against disk failures, mirroring guards against site failures, snapshots address user errors, and tape backup protects against software errors and provides archival storage.

These building blocks for data protection are proven technologies, and they continue to be improved. Even so, it is difficult to combine and configure them to create well-engineered data dependability solutions. No solution can ever provide an absolute guarantee of

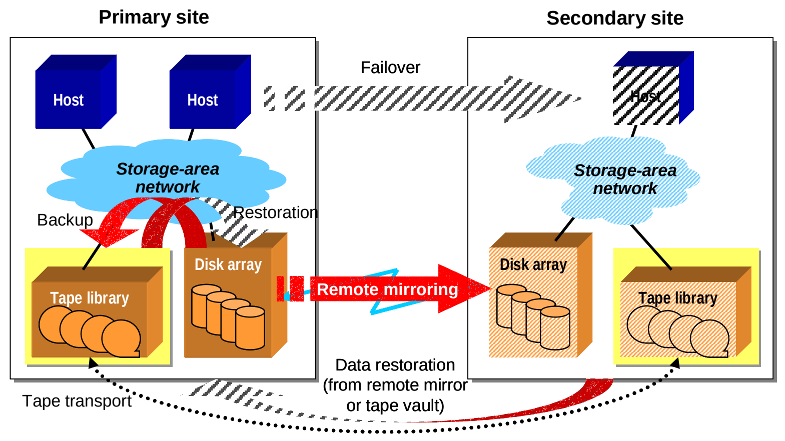

Figure 1: an overview of the disaster recovery alternatives considered in this paper. The standard solutions include inter-array mirroring (local and remote, synchronous and asynchronous), tertiary storage (e.g., tape) backup, remote vaulting, and snapshots, combined with recovery by failover or data reconstruction.

safety, so designers must trade off the desire for rapid recovery and minimum data loss against each solution's price, operational costs, and performance impact across a range of configuration choices. Over-engineered solutions may incur excessive costs to defend against negligible risks. Under-engineered solutions have their own costs in a disaster: crippling outages, loss of critical data, or unacceptably degraded service. Faced with these choices, designers often resort to *ad hoc* attempts to strike the right balance, guided by rules of thumb and their own limited (or even irrelevant) experience.

The key contribution of this paper is to show how to design data dependability solutions *automatically*, given specifications of workload properties, system components, data value, and expected rates of primary copy failures. We describe and evaluate a design tool that combines quantitative models of the standard protection and recovery techniques (depicted in Figure 1) with an off-the-shelf optimizing solver. The output of the tool is a cost-effective data dependability design.

The paper makes three further contributions, each of which is essential to automating dependability design:

1. *Dependability metrics.* We describe metrics to specify data dependability requirements, covering both data *reliability* (i.e., the absence of data loss or corruption) and data *availability* (i.e., the ability to access data when desired). These metrics characterize the severity of a failure in terms of the *financial* cost of the resulting data loss and service outages.
2. *Models of data protection alternatives.* We develop quantitative models of the dependability properties and costs of common disaster tolerance solutions.
3. *Optimization framework.* We show how to formulate data dependability design as an efficient optimization problem to balance cost and risk exposure—precisely the kind of task that automation can aid. The optimization objective is to minimize the sum of *outlays* and the financial *penalties* for failures.

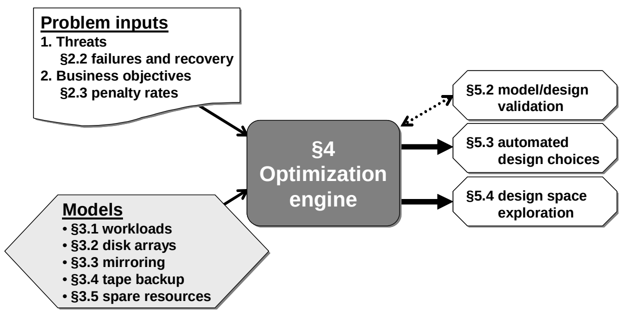

Figure 2: structure of the dependability design tool. Section numbers on the graphic indicate the sections that discuss each topic in the text.

2 Overview

To automate the design process, we must first capture the design objectives in a quantitative way. The formulation must be solution-independent, which means it specifies only the goals, and not the solutions. "Backup my data once a day" describes an implementation, rather than the underlying objective. A better choice is "lose no more than 24 hours' updates." Better still is "minimize my total costs if each hour of lost updates costs me $500,000."

In a business context, most choices come down to money. A key premise of our work is that effective dependability design must meet concrete objectives specified in financial terms. This section explains our financial framework for specifying dependability goals and configuring cost-effective solutions for data protection and recovery.

2.1 Failures and recovery

Data dependability solutions protect against several threat categories:

- *Data loss:* threats that cause data to be lost, including failures of the storage devices themselves, or an encompassing failure, such as a building fire. Recent updates may be more vulnerable to loss because they may not have propagated to fully protected storage (e.g., they might reside in a volatile OS buffer, or might not have propagated to a remote site yet).
- *Data corruption:* threats that change data into a form that cannot be used. Corruption may result from user error, defective software or firmware, and attacks.
- *Data inaccessibility:* threats that do not damage data, but prevent access to it, such as failure of a network link, switch, or disk array controller.

To limit the scope of this study, we focus on data loss events for the primary copy, such as a primary site disaster or failure of a primary disk array. Data corruption and inaccessibility threats can be mapped into loss of the primary copy. We leave a treatment of secondary failures to future work.

After a failure, normal operation resumes when either of two states is achieved:

1. *Failover:* the secondary is usable as the primary.
2. *Reconstruction:* the primary is restored and usable again, either at its original site or at another site outside of the scope of the failure.

Our tool considers only the minimum repairs needed to restore service. In practice, repair is not fully complete until the original level of redundancy is restored so the secondary can tolerate another failure. In the case of failover, typically a planned *fail-back* operation restores each site or copy to its pre-failure role.

2.2 Recovery time and recovery points

Business continuity practitioners currently use two metrics to capture data dependability objectives:

1. *Recovery time*. The *Recovery Time Objective (RTO)* specifies the maximum allowable delay until application service is restored after a failure event. The RTO can range from seconds to days.
2. *Data loss.* Recovery may require reverting to some consistent point prior to the failure, discarding updates issued after that recovery point. The *Recovery Point Objective* (RPO) gives the maximum allowable time window for which recent updates may be lost. The RPO can range from zero (no loss is tolerable) to days or weeks.

Discussions with storage designers suggest that the dependability objectives for the repair step are to restore the original level of redundancy "soon." How soon and at what expense are less well-defined: best-effort solutions are generally used.

2.3 Penalty rates

Choosing the right RTO and RPO requires understanding (1) the business costs of service outages and data loss *and* (2) the recovery behavior and cost of all available solutions to achieve each RTO and RPO. This process is a complex optimization problem in a large design space, and is hard to do well by hand.

Instead of fixed RTO and RPO targets, our approach quantifies the financial impacts of outages and data loss as *penalty rates*, expressed as \$/hour. The *loss penalty rate* specifies the cost per hour of lost updates. The *outage penalty rate* specifies the cost per hour of service interruption. Our design tool uses these rates to balance expected failure impacts against outlays to arrive at a cost-effective solution. The tool identifies the RTO and RPO as a *side effect* of this design process.

Determining penalty rates is still not easy, but it can be done using techniques such as Business Impact Analysis (BIA), and tools (e.g., http://www.availability.com) that help managers quantify their exposure to outages and data loss. The insurance industry regularly assesses threats and exposures in many other domains of equivalent complexity. Assessing these penalty rates is a strictly simpler and less error-prone task than determining the right RTO and RPO values.

Even though site and regional failures are extremely rare in North America, government regulation or liability obligations [CNT2003] may mandate specific maximum RPO and RTO values. More stringent requirements are particularly likely in the financial industry [SEC2002]. Mandated values can be specified as bounds for the RPO and RTO results obtained from our approach, although the bounds are rarely tight.

2.4 Putting it all together

Our design tool works as follows. First, it uses the models described in Section 3 to explore the solution space—including the ranges of configuration parameter settings for each alternative—and to predict the deployment outlays, worst-case recovery time, and worst-case data loss for each candidate solution. The tool then selects the best design alternative (e.g., synchronous mirroring), together with the best values for its configuration parameters (e.g., two wide area links) and the best recovery alternative (e.g., failover to the secondary site). This optimal configuration balances the system cost (outlays) with the worst-case financial costs (penalties) for expected primary failure events.

3 Modeling protection techniques

This section outlines the models and parameters for the data protection and recovery alternatives considered in this paper. Each model defines a set of input parameters (e.g., component outlays and performance attributes) and their values, a set of configuration parameters (e.g., number of tape drives or network links to a remote mirror) and their ranges of valid settings, and equations relating the parameters to the measures of interest: worst-case data loss and recovery time. It is trivial to model different parameter settings for system components, and it is straightforward in principle to incorporate new data protection techniques into the design tool.

To normalize costs, all capital outlays are depreciated linearly over three years, and all recurring costs (e.g., leasing and support costs) are given at an annual rate.

3.1 Workload characteristics

Data dependability designs are sensitive to characteristics of the storage workload. Many data protection schemes are sensitive to the *average update rate*: the volume of updates over a given interval, divided by the interval length. Techniques that accumulate modifications over an interval (e.g., incremental tape backup) are more sensitive to the workload's *unique update rate*, the number of unique data items written over an interval. Longer accumulation intervals allow more time for overwrites, so they often have lower unique update rates. We model this rate by a series of tuples of the form <interval duration, unique update rate>. Synchronous mirroring solutions are also sensitive to the short-term peak-to-average burstiness of writes, which is typically 3–10× the long-term average update rate.

These characteristics can be measured from an existing system or estimated by a person or a tool from a repertoire of well-known workloads. Table 1 quantifies the relevant workload characteristics for the cello2002 workload [Ji2003], a publicly available trace of a time-sharing system at HP Labs. The experiments reported in Section 5 use this workload.

Parameter	Description	Values
wkldCapacity	overall workload capacity	1.36 TB
avgUpdateRate	average update rate; no rewrite absorption	799 KiB/s
burstMultiplier	short-term burst update-rate multiplier	10×
uniqueUpdate-Rate(duration)	unique update rate over the given interval, after absorbing overwrites: <interval duration, unique update rate>	<1 min, 727KiB/s> <5 min, 689KiB/s> <1 hr, 581KiB/s> <4 hr, 458KiB/s> <12 hr, 350KiB/s> <24 hr, 317KiB/s> <48 hr, 317KiB/s>
uniqueCapacity (duration)	total size (capacity) of unique updates during given duration	*duration × uniqueUpdate-Rate(duration)*

Table 1: workload attributes for cello2002.

We ignore workload characteristics that do not affect the choice of dependability solutions. Existing tools can address these design issues (e.g., [Anderson2002a]).

3.2 The primary copy

We assume one or more disk arrays store the primary copy of data. We assume that they protect data against internal single-component failures, and consider only complete failure of the primary array or site. For simplicity, we consider the case where the entire dataset is protected in the same way; in practice, different storage volumes may be protected differently.

Our disk array cost model is based on a diverse set of disk arrays from Hewlett-Packard, including the HSG80, EVA and XP1024 arrays. The model captures details such as the costs of the array chassis/enclosures, redundant front-end controllers (including caches), XP1024 back-end controllers, and the disk drives and the trays in which they are mounted. It estimates the cost of floor space, power, cooling, and operations by using a fixed facilities cost plus a variable cost that scales with capacity. For the experiments reported here we used RAID-10 configurations of the EVA model summarized in Table 2.

Parameter	Description	Values
maxDisks	upper bound on number of disks in each array	256
diskCapacity	capacity per disk drive	73 GB
arrayCacheCapacity	array cache capacity	32 GiB
arrayReloadBW	maximum disk array reload (restore) rate	512 MB/s
enclosureCost	outlay cost of disk array enclosure	$189 890 / enclosure
diskCost	outlay cost of disk	$3549 / disk
fixedFacilitiesCost	fixed outlay cost	$60k / year
varFacilitiesCost	variable outlay cost	$1 / GB / year
depreciationPeriod	amortization period for capital outlay costs	3 years

Table 2: primary copy model parameters for the HP EVA disk array and facility costs. These EVA costs were representative at the end of 2003, and may not be actual list prices. The facility costs are estimates.

The overall annualized outlay cost for disk array storage and facilities for the primary copy is:

$$primaryCost = \frac{\begin{pmatrix} numDiskArrays \times enclosureCost \\ + numDisks \times diskCost \end{pmatrix}}{depreciationPeriod} + varFacilitiesCost \times wkldCapacity + fixedFacilitiesCost$$

Parameter	Description	Values
mirrorCache-Capacity	size of buffer used to smooth update bursts	100 MiB
$interval_{asyncB}$	asyncB batch duration for coalescing writes	1 min, 5 min, 1 hr, 4 hr, 12 hr, 24 hr
linkBW	link bandwidth	T3: 6MiB/s OC3: 16MiB/s
$links_{max}$	upper bound on number of links	16
linkCost	annual outlay cost for link	T3: $60 000 / year OC3: $456 000 / year

Table 3: parameters for the remote mirroring model. Link cost estimates come from [Ji2003].

3.3 Remote mirroring

Remote mirroring protects against loss of the primary by keeping an isolated copy on one or more disk arrays at a secondary site. Our remote mirroring model includes a transfer rate (bytes/s) for the network link connecting the mirrors, a link cost ($/year), and an upper bound on the number of links that may be deployed. We currently model T3, OC3, OC48, and local and metropolitan-distance (10km) optical FibreChannel links. Adding new link types is as simple as specifying their parameter values. Table 3 shows the parameters for the T3 and OC3 link types used in the experiments.

The primary cost factors for remote mirroring systems are the costs of (1) the storage for the remote copy and (2) the network links required to match the write rate at the primary. The costs and penalties depend on the remote mirroring protocol [Ji2003]:

- *Synchronous mirroring (sync)*: the secondary receives and applies each write before the write completes at the primary. This scheme requires low latency (e.g., close proximity) between the sites to obtain good performance, but no data is lost if the primary fails: $dataLoss = 0$. Links are provisioned to support the short-term burst write bandwidth:

$$links_{min} = \left\lceil \frac{avgUpdateRate \times burstMultiplier}{linkBW} \right\rceil$$

- *Write-order preserving asynchronous mirroring (async)*: this conservative protocol propagates all primary writes (without coalescing rewrites) to the secondary as fast as the inter-array interconnect allows. Updates are applied in the same order at both sites, but updates to the secondary may lag. This asynchrony can improve the performance of the foreground workload beyond inter-site distances of a few tens of km, but updates may be lost if the primary fails. We configure the primary with a write buffer that is large enough to smooth the observed worst-case update bursts for the workload. As a result, the links are provisioned to support the long-term average (non-unique) update rate:

$$links_{min} = \left\lceil \frac{avgUpdateRate}{linkBW} \right\rceil$$

  If the primary fails, then any updates buffered in the write buffer are lost. The worst-case data loss window is given by the time to fill or drain the buffer:

$$dataLoss = \frac{mirrorCacheCapacity \times numDiskArrays}{min(avgUpdateRate, (numLinks \times linkBW))}$$

- *Batched asynchronous mirroring with write absorption (asyncB)*: this protocol reduces bandwidth costs by coalescing repeated writes to the same data. Updates accumulate into batches at the primary and periodically propagate to the secondary mirror, which applies each update batch atomically [Patterson2002, Ji2003]. We declare batch boundaries at fixed time intervals ($interval_{asyncB}$) ranging from one minute to 24 hours. The link bandwidth must support the worst-case unique update rate over the batch interval:

$$links_{min} = \left\lceil \frac{uniqueUpdateRate(interval_{asyncB})}{linkBW} \right\rceil$$

  The potential data loss is the size of two delayed batches (one accumulating and one in transit to the secondary), so we approximate the worst-case loss window as twice the batch interval:

  $dataLoss = 2 \times interval_{asyncB}$

The overall annualized outlay cost for mirroring is just the cost of the secondary copy plus the links:

$mirrorCost = secondaryCost + numLinks \times linkCost$

$secondaryCost = primaryCost$

Failover to a remote mirror and active standby host resources causes a short, temporary outage (30 seconds in our model). Reconstructing the entire data set on the primary from the secondary across the network takes longer:

$recoveryTime = wkldCapacity / (linkBW \times numLinks)$

The time to restore an array's worth of data after an array failure is calculated in a similar fashion, but scaled by *numDiskArrays*.

Both failover and reconstruction alternatives may require spare storage and/or computational resources at the secondary site. These may impose additional costs and/or recovery delays (see Section 3.5).

3.4 Tape backup

Table 4 summarizes the parameters of the tape backup model. Backups occur at fixed intervals ranging from 4 to 48 hours. Periodic full backups are optionally interspersed with cumulative incremental backups, which copy only the data modified since the last full backup. For example, backup intervals of 24 hours with incremental cycle counts of 6, 13 or 27 days correspond roughly to weekly, bi-weekly, or monthly full backups interspersed with daily incremental backups.

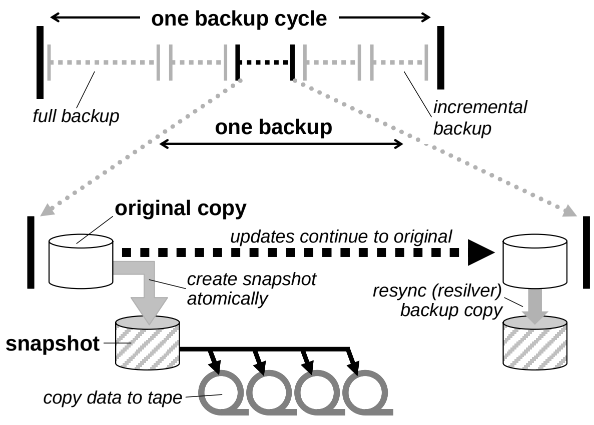

Figure 3: the backup cycle.

Parameter	Description	Values
tapeCapacity	tape media capacity (B)	SDLT: 320 GB LTO: 400 GB
tapeDriveBW	tape drive rate (B/sec)	SDLT: 16 MB/s LTO: 60 MB/s
$tapeDrives_{max}$	upper bound on number of drives in library	16
$tapes_{max}$	upper bound on number of tapes in library	600
$interval_{full}$	interval for full backups	4 hr, 12 hr, 24 hr, 48 hr
$interval_{incr}$	interval for incremental backups	4 hr, 12 hr, 24 hr, 48 hr
cycleCount	number of incrementals between full backups	0, 6, 13, 27
$interval_{cycle}$	interval between full backups	$= interval_{full} + cycleCount \times interval_{incr}$
$retrievalTime_{vault}$	time to retrieve tapes from offsite vault	1 hr
tapeLibraryCost	outlay cost for tape library, including chassis plus media slots	$148 342 per library
tapeDriveCost	outlay cost for tape drive	SDLT, LTO: $19 554 / drive
tapeCost	outlay cost for tape media cartridge	SDLT: $125 LTO: $150
fixedVaultCost	outlay for tape vault	$25 000 / year
vaultPer-ShipmentCost	outlay cost for a shipment to tape vault	$50 / shipment

Table 4: summary of tape backup parameters.

Figure 3 illustrates the backup process. It creates a consistent, read-only snapshot of the primary data, and then uses the snapshot as the source for the backup to tape (be it full or incremental). Snapshots may be taken using space-efficient copy-on-write techniques (e.g., [Patterson2002, Lee1996]), or by isolating a local mirror and synchronizing it with the primary copy after the backup is complete (e.g., [EMC–SRDF, HP–XP–CA]). The disk space required for a space-efficient incremental snapshot is determined from the average unique update rate and the backup interval.

Each backup must finish before the next one starts, effectively defining a *backup window* equal to the interval duration. The tool provisions sufficient tape drives to complete each backup within its window:

$$tapeDrives_{min} = \max(tapeDrives_{minFull}, tapeDrives_{minIncr})$$

where the number of drives needed for a full backup is:

$$tapeDrives_{minFull} = \left\lceil \frac{wkldCapacity}{interval_{full} \times tapeDriveBW} \right\rceil$$

and the number of tape drives required for the largest incremental backup is:

$$tapeDrives_{minIncr} = \left\lceil \frac{(cycleCount - 1) \times uniqueCapacity(interval_{incr})}{interval_{incr} \times tapeDriveBW} \right\rceil$$

Tapes are retained for a single full backup cycle, which includes the last full backup and all subsequent incremental backups. Each full backup is written onto a new set of tapes rather than the tapes for the previous full backup, in case it fails to complete. When a full backup completes, the tapes for the previous full backup are sent to the vault, and the tapes at the vault are recycled back to the primary site. The tapes are kept at the primary until this time in case they are needed quickly to respond to operator errors (e.g., rm *). Thus the total number of retained tapes is:

$$numTapes = 2 \times numTapes_{full} + numTapes_{incr}$$

where the number of tapes required for a full backup is:

$$numTapes_{full} = \lceil wkldCapacity / tapeCapacity \rceil$$

The number of tapes required for all incremental backups during a cycle is calculated by summing the number of tapes used for each one. We assume that each backup starts on a new tape. Taking into account the fact that the full backup interval may be larger than the incremental one, we get:

$$numTapes_{incr} = \sum_{i=0}^{cycleCount-1} \left\lceil sizeOfIncr(i) \middle/ tapeCapacity \right\rceil$$
$$sizeOfIncr(i) = uniqueCapacity(interval_{full}) + i \times uniqueCapacity(interval_{incr})$$

Tape libraries hold both tape drives and tape cartridges. We model HP's series ESL9595 libraries, which can handle 2 to 16 drives, up to 600 cartridges, and three different tape cartridge/drive technologies (DLT, SDLT and LTO). The number of tape libraries needed to house the tapes and tape drives is:

$$numTapeLibraries = \max\left(\left\lceil \frac{numTapeDrives}{tapeDrives_{max}} \right\rceil, \left\lceil \frac{numTapes}{tapes_{max}} \right\rceil \right)$$

We assume that tapes are replaced every year, to guard against media failures. Thus, the minimum annualized outlay cost for backup is:

$$backupCost = \frac{\begin{pmatrix} numTapeLibraries \times tapeLibraryCost + \\ numTapeDrives \times tapeDriveCost \end{pmatrix}}{depreciationPeriod} + numTapes \times tapeCost$$

To protect against site disasters we add a second tape library at the reconstruction site. Disaster protection also incurs an annual cost for tape vaulting:

$$backupCost = 2 \times \frac{\begin{pmatrix} numTapeLibraries \times tapeLibraryCost + \\ numTapeDrives \times tapeDriveCost \end{pmatrix}}{depreciationPeriod}$$
$$+ numTapes \times tapeCost$$
$$+ fixedVaultCost$$
$$+ vaultPerShipmentCost \times shipmentsPerYear$$

3.4.1 Data loss

A primary array failure may destroy any backup in progress at the time of the failure, possibly losing all updates from both the current (accumulating) backup interval and the previous (propagating) backup interval. Assuming full intervals are at least as long as incremental intervals, the worst-case data loss window is the sum of the full and incremental backup intervals:

$$dataLoss = interval_{full} + interval_{incr}$$

In the event of a primary site disaster, the worst-case data loss occurs if the site is destroyed just before the new full backup completes and the old full backup is shipped offsite. In this case, the data at the vault is out-of-date by twice the full backup cycle duration, plus the interval for the latest full backup:

$$dataLoss = 2 \times interval_{cycle} + interval_{full}$$

3.4.2 Recovery time

Recovery from tape is a three-phase process: first, if the tapes are stored at an offsite vault, they must be retrieved to the reconstruction site; second, the latest full backup is restored and third, the latest subsequent incremental backup is restored. Vaults can be close to or far away from the target reconstruction site. The largest capacity incremental backup is the last one of the cycle. We make the following simplifying assumptions: all the tape drives in each library operate in parallel during each phase, and data is spread evenly across the tapes and drives. We ignore tape load time because it is typically less than 5% of the time to read the tape.

The number of tapes for the last incremental backup is:

$$numTapes_{maxIncr} = \left\lceil \frac{(cycleCount - 1) \times uniqueCapacity\left(interval_{incr}\right)}{tapeCapacity} \right\rceil$$

For a site disaster, the worst-case recovery time is the time to retrieve the tapes from the offsite vault, plus the time to restore the last full and the last incremental backup of a cycle:

$$recoveryTime = retrievalTime_{vault} + recoveryTime_{full} + recoveryTime_{incr}$$

$$recoveryTime_{full} = \frac{tapeCapacity \times numTapes_{full}}{tapeDriveBW \times numTapeDrives}$$

$$recoveryTime_{incr} = \frac{tapeCapacity \times numTapes_{maxIncr}}{tapeDriveBW \times numTapeDrives}$$

Recovery after an array failure is analogous; the main differences are that the tapes are not yet offsite (e.g., zero $retrievalTime_{vault}$), and the tape counts for the full and last incremental backups are scaled by *numDiskArrays*.

3.5 Spare resources

After a failure, recovery can begin immediately if hot standby resources are available. Otherwise, resources must be found or acquired, drained if they are in use for another purpose, and (re)configured or (re)initialized if necessary. Solutions may minimize recovery time by keeping spare equipment in various states of readiness.

Failover or reconstruction at the secondary site requires computational resources at the target site. Since our focus is on storage system design, our tool always provisions hot spare server(s) at the target site if it selects one of these recovery alternatives. For simplicity we set the cost equal to the outlay cost of the primary disk arrays for the workload, following the rule of thumb that storage and servers each account for one third of the cost of a typical server site.

For reconstruction alternatives, we consider only reconstruction at the secondary site or at the primary over the same interconnect provisioned for backup or mirroring. Reconstruction at a third site would require additional, independently provisioned network links.

Reconstruction requires one or more disk arrays to serve as target for the restoration. We model a spectrum of spare resource options by the outlay cost of maintaining them and the time to provision and configure them. Table 5 gives the model parameters, and Figure 4 illustrates the alternatives. In all cases, we assume that any spare resources consumed by recovery are eventually replaced with new equipment, and factor the replacement equipment cost out of the equations.

Parameter	Description	Values
spareCost	outlay cost of spare disk array storage and facilities	*=primaryCost*
spareDiscount	discount factor for shared resources (fraction in [0,1])	0.2
t_{order}	time to order, deliver, and set up new resources	24 hr
$t_{identify}$	time to identify that resources are available	60 sec
$t_{configure}$	time to configure resources	10 hr
t_{scrub}	time to scrub resources	5 hr
$t_{negotiate}$	time to negotiate the release of shared resources	4 hr

Table 5: model parameters for spare resources. Spare costs include the cost of the entire target site for site disasters; for array failures, costs include only the array and related capacity-dependent facilities costs.

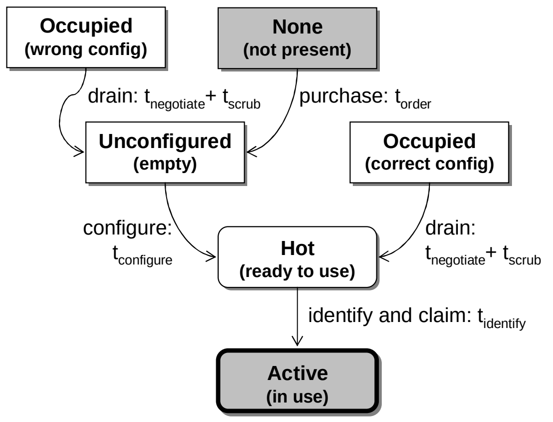
Occupied (wrong config)
None (not present)
drain: $t_{negotiate} + t_{scrub}$
purchase: t_{order}
Unconfigured (empty)
Occupied (correct config)
configure: $t_{configure}$
Hot (ready to use)
drain: $t_{negotiate} + t_{scrub}$
identify and claim: $t_{identify}$
Active (in use)

Figure 4: spare resource alternatives, and their preparation times. The time to provision spare resources is calculated by summing the times along the path from the chosen alternative to the active state.

One way of gaining access to spare resources is to rent access to a shared resource pool. Several companies offer such a service, which can be much cheaper than a dedicated backup site. We model the cost of shared resources by a fixed discount factor (a fraction in [0, 1]). However, the lower cost comes at a risk:

> *"During the days following September 11, some disaster recovery vendors found they were unable to accommodate all of their affected clients, with the result that several institutions found themselves without the anticipated backup facilities."* *[SEC2002]*

4 Automating dependability design

Our automated design tool, or *solver*, uses the data protection and recovery models described in the previous section to determine the most cost-effective solution for a given set of workload inputs, component parameters, assumed rates of primary array failures and primary site disasters, and failure penalty rates. Outputs from the solver include the choice of design alternative, the worst-case data loss window and recovery time (RPO and RTO), and the outlays and expected penalties. The solution also specifies settings for configuration parameters such as the number of tape drives, backup intervals, update batch interval, and other model parameters discussed in Section 3.

4.1 The optimization problem

We based our solver on mathematical programming primarily because of its expressive power and efficiency in traversing a large and complex search space. The technique has the added advantage that it arrives at a mathematically optimal solution; however, because the failure penalty rates are merely estimates, this fact is less important. Because our emphasis is on storage system design and not optimization technology, we used an off-the-shelf optimization tool, rather than develop our own. We cast the problem as a mixed-integer optimization with constraints. Our prototype is written in the GAMS language for the CPLEX solver [GAMS1998, ILOG2002].

The solver's objective is to minimize overall business cost, defined as outlays plus failure penalties. Within the solver, a set of binary decision variables represents the data protection alternatives and their base configurations. Each binary variable corresponds to a single protection alternative (e.g., mirroring or backup) and a specific set of discrete configuration parameters (e.g., "batched asynchronous mirroring with a write absorption interval of one minute"). A second set of binary decision variables represents the alternatives for recovery and spare resources (e.g., "use failover"). Integer decision variables represent the number of bandwidth devices (e.g., network links or tape drives) for each protection alternative.

The solver uses the following optimization constraints:

- Exactly one protection alternative must be chosen.
- Exactly one recovery alternative must be chosen.
- The number of bandwidth devices for a data protection alternative is either zero (if that alternative has not been chosen), or it is within the range specified by the upper and lower bounds calculated or specified for the alternative (e.g., $[links_{min}, links_{max}]$).
- The aggregate device bandwidth may not exceed the aggregate reload rate for the primary disk array(s).

4.2 Optimizing the optimization

The multitude of parameter combinations leads to a complex search space. We applied several techniques to make the solver more efficient. With these improvements, the optimization runs in just a few seconds.

First, we reduced the search space considerably by quantizing the values of certain continuous parameters, such as backup intervals and the batch intervals for the batched asynchronous remote-mirroring scheme.

Second, we divided several continuous, non-linear functions to a set of piecewise-linear segments. When formulated in the straightforward fashion described above, the recovery time models have terms that depend inversely on the number of links or tape drives *y*. The resulting optimization problem becomes non-linear. Although solvers exist for certain classes of non-linear optimization problems, they may take an unacceptably long time to find a solution, or fail to find one at all. Linear solvers exploit well-known theoretical results about the search space structure to solve significantly larger problems in seconds. To linearize our problem, we defined new variables z standing for the inverse of the problematic y terms. We introduced new constraints on the variables so that the resulting optimization problem is equivalent to the original one. Since the transformation is convex, we used the methods described in [Williams1999], which linearize convex constraints in continuous variables. If the variables had been continuous, this would have been an approximation, but because our decision variables are integers, we can prove that the result is exact.

5 Experimental results

We conducted a series of experiments to explore the behavior of our automated dependability design tool:

1. To validate the solver's recovery time, data loss and cost calculations, we evaluated these quantities for a fixed set of designs.
2. To explore the solver's design choices, we supplied a set of penalty inputs corresponding to specific industry segments, and examined the output designs.
3. To explore the sensitivity of the solver's design choices to its inputs, we varied the penalty rates across a wide range, and examined the outputs.

Although workload parameters do affect the configuration parameters for each solution, our experience has shown that the penalty rates dominate the design choices. We use a single workload for all experiments (cello2002, described in Table 1). All experiments assume a failure rate of one primary site disaster per year. The effect of changing the failure frequency is equivalent to scaling the penalty rate values by an appropriate multiplier; the experiments explore in detail how the design choices change as these penalty rates vary.

5.1 Validation of fixed designs

To validate our intuitions about the recovery time, data loss and cost behaviors of candidate designs, we ran the solver in *evaluation* mode. In this mode, the solver's decision variables are fixed to a particular design choice and configuration parameter values. We present a representative sampling of the results here, all of which were set up to reconstruct the data to a hot spare at the primary site in the event of a site disaster.

Figure 5 presents a set of results for tape backup. The graph shows recovery time as a function of the number of tape drives for two different tape drive technologies (SDLT and LTO). We considered two backup schedules: one using just full backups with a four-hour backup interval, and one using weekly full backups plus daily incremental ones, both with 24-hour intervals. The graphs for the full-only policy begin at two drives (for LTO) and seven drives (for SDLT) because that is the minimum number of tape drives required to back up the entire data set in the four-hour window. For LTO, the curves stop at eight drives because the aggregate tape drive bandwidth beyond this point is larger than the array reload rate, so additional drives do not help.

As expected, the slower SDLT technology results in longer recovery times. For both technologies, recovery is faster with the full-only schedule: it restores only the last full backup, while the weekly full/daily incremental schedule must also apply the latest incremental backup.

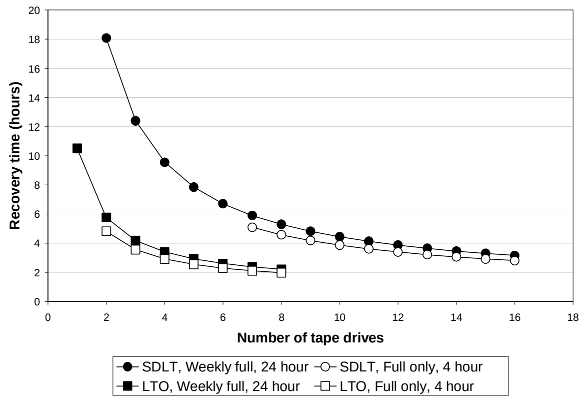

Figure 5: tape backup recovery time as a function of tape drive/cartridge technology and backup policy.

The graph confirms the importance of provisioning to minimize recovery time in this scenario: for smaller numbers of tape drives, the outage penalty reduction from adding a tape drive exceeds the drive's purchase price. The LTO drives are particularly cost-effective.

Protection technique	Recent data loss
Synchronous mirroring	0
Asynchronous mirroring	2.2 min
Asynchronous batch mirroring (interval = 1 min)	2.0 min
Asynchronous batch mirroring (interval = 1 hour)	2.0 hours
Backup (full-only; interval = 4 hours)	12 hours
Backup (weekly full + daily incremental; both intervals = 24 hours)	360 hours

Table 6: recent data loss for different designs.

Table 6 summarizes the worst-case data loss for several design alternatives. Synchronous mirroring experiences zero data loss. For non-coalescing asynchronous mirroring, data loss corresponds to the time for the workload to fill the 100 MB mirroring write cache. Data loss for the asynchronous batch variants is twice the batch duration. The worst-case data loss for tape backup and vaulting is twice the backup cycle, plus the backup interval for the latest full backup. The large values show the down-side of keeping tapes on-site rather than sending them to the vault in the case of site failure.

Figure 6 shows how outlay and expected penalty costs combine to yield the total cost of a candidate solution across a range of configuration parameters. The graph shows outlay and penalty costs for remote asynchronous mirroring with recovery by reconstruction at the primary. This experiment assumes the penalty rate inputs for both data loss and outages are $20 000 per hour.

Outlay costs increase linearly as the number of configured T3 network links increases. The expected data loss for each failure is constant with this solution, but adding more links speeds recovery by increasing the bandwidth from the mirror to the primary. The graph shows a "sweet-spot" that minimizes the overall cost at five T3 links: with fewer links, slower recovery drives up the outage penalties, while adding more links yields diminishing benefit and causes link outlays to dominate. We ran this experiment in evaluation mode for illustrative purposes; in design mode the goal of the solver is to find such sweet spots automatically.

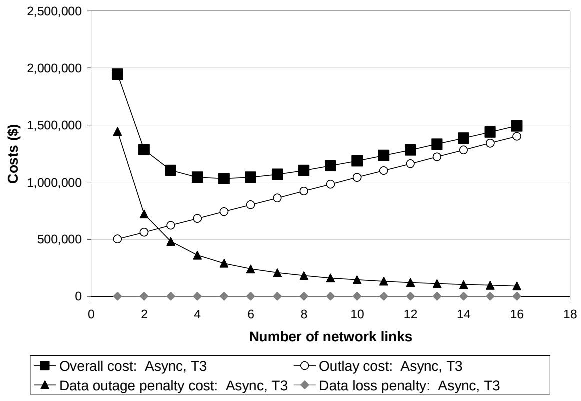

Figure 6: outlay and penalty costs for asynchronous mirroring as a function of link configuration.

5.2 Automated design choices

For the second set of experiments we ran the tool in design mode to configure a data protection system for one primary site disaster per year. These experiments explore the design choices for penalty rates that are typical of specific industry segments [CNT2003b]. Table 7 summarizes the data loss and outage penalty inputs for each industry segment, together with the chosen design and its cost. Figure 7 illustrates the cost breakdown for each solution.

Type	Data loss penalty	Outage penalty	Design chosen	Overall cost
Student accounts: Storage of data owned by students is tolerant of data loss and outages.				
	$500 / hr	$500 / hr	backup with 12-hr interval + no spares	$301k
Company documents: Documents such as papers, presentations, and design documents are tolerant to small outages and loss of recent writes.				
	$500k / hr	$500 / hr	async mirror+ reconstruction + no spares	$426k
Web server for an online retailer: Outages are expensive, because if the web server goes down, orders stop. Data loss is tolerable, because data can be replaced from other sources.				
	$500 / hr	$500k / hr	asyncB mirror + failover	$506k
Consumer banking: Consumer banking services are tolerant of modest outages (since account holders are unlikely to switch banks), but not data loss.				
	$50M / hr	$50k / hr	sync mirror + failover	$562k
Central bank: Central banks are required by regulations to have zero data loss and small outage windows.				
	$50M / hr	$5M / hr	sync mirror + failover	$603k

Table 7: summary of industry segment examples.

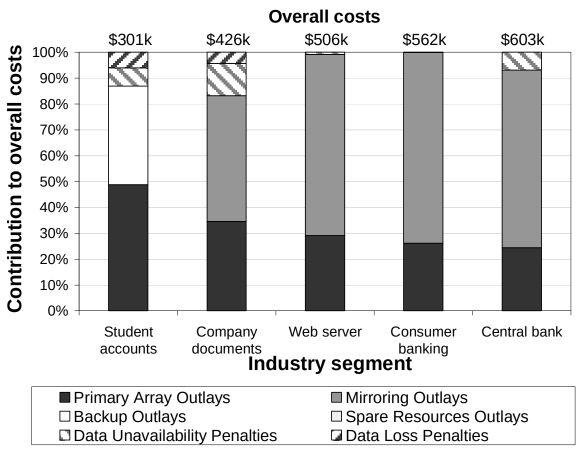

Figure 7: solution costs by industry segment.

For lower penalty rates (student accounts and company documents), the tool chooses designs that employ reconstruction with no spare resources: the outlay costs for better protection exceed the failure penalties. It switches to asynchronous mirroring and failover at moderate penalty rates (web server), because the outlays for this stronger solution are less than the penalties expected with a cheaper solution. At the highest penalty rates (consumer banking and central bank), the tool selects an aggressive synchronous mirroring + failover solution to achieve zero data loss and minimal outage time. Even so, the mission-critical central bank incurs a non-trivial outage penalty for the 30-second failover time.

Interestingly, the designer always chooses a solution whose expected penalties are a modest share of the overall cost (at most 17%).

5.3 Solver design choice sensitivity

Our final set of experiments maps the solution space for data protection and recovery across a range of failure penalty rates. We performed over 500 separate experiments, varying the data loss and data outage penalty rates independently over a wide range (10^7:10^1). Exploring this space by hand using existing tools would be unthinkable. Our intent is to give insight into the richness of the design space, to show the feasibility of navigating this design space automatically, and to provide further evidence that the design tool is correct and efficient.

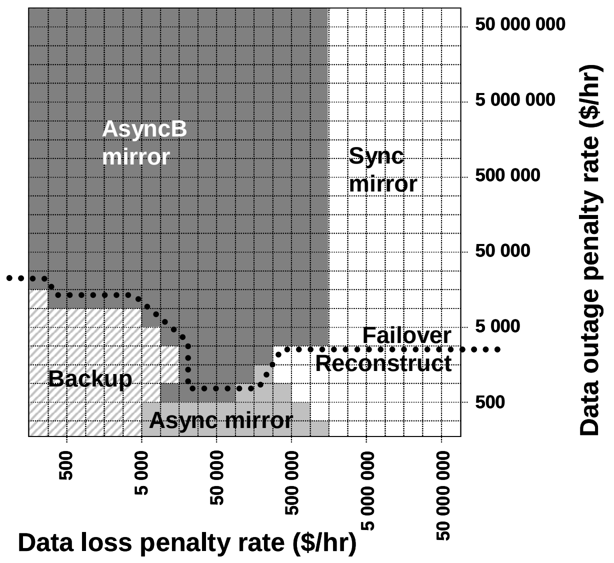

Figure 8: solution types picked by the solver for the space of penalty rates. The dotted line separates failover and reconstruction solutions.

Figure 8 shows the design alternatives selected for each input scenario. As expected, the tool chooses the cheapest solution—tape backup—for scenarios tolerant of outages and data loss (e.g., student accounts), but incorporates hot spare resources and more aggressive backup intervals as the penalties for outage and data loss increase. The tool chooses asynchronous batch mirroring for the scenarios with more severe outage penalties when the cost to lose recent writes is acceptable (e.g., web server). At low data outage penalty rates, as the penalty rate for data loss continues to increase (e.g., company documents), the solutions switch to order-preserving asynchronous mirroring to reduce the data loss beyond what tape backup can offer. As the penalty rates for data loss increase further (e.g., consumer banking and central bank), the solutions shift to synchronous mirroring.

When outage penalties are low, the mirroring solutions use reconstruction rather than failover to avoid incurring the additional outlays for standby compute resources. As outage penalties increase, the solutions incorporate hot spare standby resources and failover to minimize downtime.

To clarify the reasoning behind the solver's decisions, we now look at its calculations in greater detail. Figure 9 and Figure 10 show the recovery time and recent data loss, respectively, for the chosen solutions.

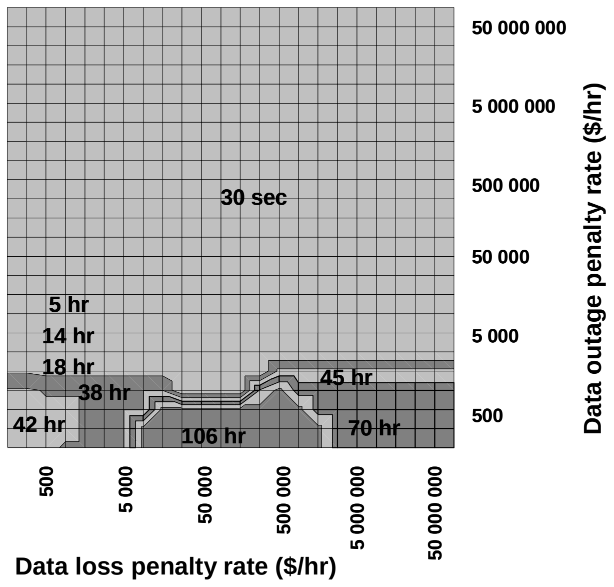

Figure 9: recovery time as a function of penalty rates. Each tinted band corresponds to a 20–hour increase in recovery time.

For the lowest data loss and outage penalty rates, the solver chooses tape backup with no spare resources, full-only backups and twelve-hour intervals, resulting in a recovery time of 42 hours and data loss of 36 hours. As the data outage penalty increases, the backup-based solutions shift to using occupied spare resources and four-hour backup intervals, decreasing recovery time to 14 hours. The reductions in backup intervals also reduce the data loss window to 12 hours. Adding hot standby resources further reduces recovery time to five hours. The shift to asynchronous batch mirroring + failover at high outage penalty rates dramatically reduces the recovery time to 30 seconds and the data loss window to two minutes.

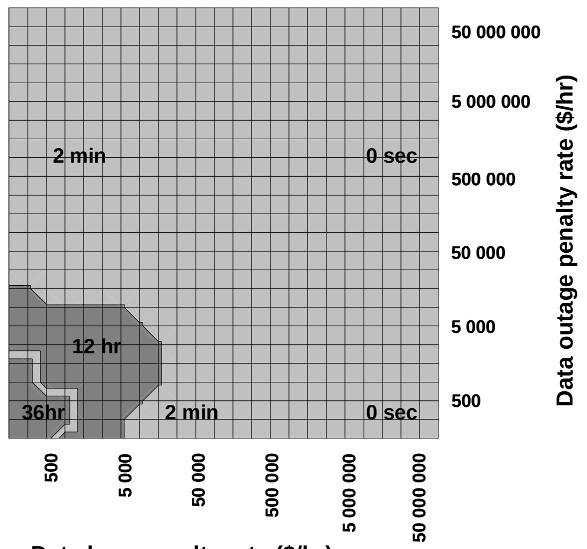

Figure 10: data loss time as a function of penalty rates. Each tinted band corresponds to a 10-hour increase in data loss time.

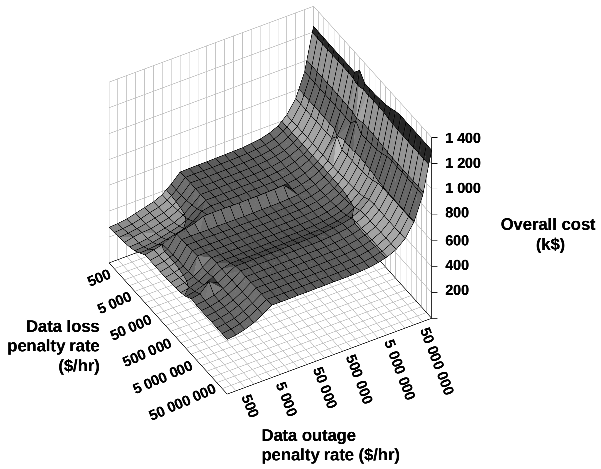

Figure 11: total (penalty+outlay) costs (k$) for the space of penalty rates.

For asynchronous mirroring with low outage penalties, the tool selects reconstruction rather than failover; it tolerates 100+ hour outage times, rather than incur the expense of standby compute resources for failover. As the loss penalty rate increases, the shift to synchronous mirroring + reconstruction reduces data loss to zero, with the added benefit of decreasing the recovery time to about 70 hours. (This reduction comes from the additional networking links synchronous mirroring requires to support update bursts.) As the outage penalty rates increase, the synchronous mirroring solutions shift to failover, resulting in recovery times of 30 seconds.

Figure 11 shows the overall costs associated with the solver's choices. In all cases the tool balances outlays with penalties, increasing system cost only as needed to avoid larger penalties. One notable crossover occurs as more aggressive spare resource options are purchased (from *none* to *occupied* to *hot spare* to *failover*), to reduce the outage penalties. Another interesting crossover occurs as the choice of protection technique shifts (from more aggressive backup policies to asynchronous mirroring to synchronous mirroring) to reduce data loss penalties.

5.4 Discussion

Asynchronous mirroring performed surprisingly well in this study. We were initially skeptical, but found that the additional high costs of high-speed, long distance links largely relegated synchronous mirroring to situations where the small data loss that could result with asynchronous mirroring is incredibly expensive – as it is, for example, in the central bank case, where a single transfer can involve millions or even billions of dollars.

We noted that the solver generally chose to use full-only backups, rather than interspersing incremental backups between full ones. Even though the tape costs for incremental backups are generally lower, this outlay savings is outweighed by the increased recovery time penalties that result from the need to restore both the latest full and latest incremental backups.

We observed firsthand that even people who should know better can make mistakes when faced with this rich design space. For example, we were experimenting with an early version of the design tool, exploring the effects of increasing the outage penalty rate for a mirroring + reconstruction solution to protect against array failure, when we noticed that the tool chose to configure a local tape library at the primary site. Why? Because we had constrained mirroring in that version to provision only enough networks links to absorb the update traffic for asynchronous mirroring. Unfortunately, the time to reconstruct the primary over these economical, slow links was so large that it became cost-effective to restore from a local tape library, rather than trickle the data back from the remote mirror. The original design was wrong: the solution should have been designed to handle the recovery case, not just the mirror update case. We should not have made this mistake, but we did. The automated dependability design tool can help to avoid this type of mistake.

6 Related work

We found very few other systems and tools that automate the design of dependability solutions. Most of the tools we found tackle only pieces of the problem, and rely on people to do the rest.

Storage vendors, such as IBM, Sun, HP, Computer Associates and Exabyte, provide web-based and downloadable tools to help with low-level backup provisioning questions, and to estimate total cost of ownership (TCO) and return on IT investments (ROI). For example, Sun's backup calculator [Sun2004] computes the number of libraries, tape drives and tape cartridges required to support a backup schedule whose parameters are supplied as inputs. This type of tool provides no indication of the dependability of the resulting backup system, nor any indication of the financial ramifications of its use.

ROI and TCO calculators (e.g., [EMC2004]) provide estimates for the total cost of ownership for alternative storage systems, given inputs for storage capacity, utilization and availability. These tools incorporate outlay costs for storage and server hardware and administrator salaries, and penalty costs for outages (but not data loss). Their overall system costs are based on input values for system availability, rather than calculated values for the dependability of a storage system configuration.

These tools provide simplistic evaluations of low-level configuration parameters or overall system costs, respectively. By changing input parameters, one can perform a rudimentary sensitivity analysis. Neither class of tools can propose a storage system configuration to meet user-specified dependability goals.

Like many other researchers in the dependability community, we build models of the systems we are trying to design. Our work emphasizes embedding those models in an automated design system. Most other work, however, focuses on models and modeling toolkits that allow designers to interactively try "what-if" analyses by hand (e.g., [Deavours2002, Haverkort2001]).

System designers and analysts make the design choices we describe here every day, but the focus in their literature is on operational issues, such as determining backup policies and setting related configuration parameters (e.g., [Cougias2003, Marcus2003, Preston1998, daSilva1993]).

The majority of "data protection" work is focused on developing new protection mechanisms. Only rarely do their designers consider the difficulties of deciding when to use them, and how they combine with existing techniques [Wylie2001].

Researchers have developed automated tools to design systems to meet performance goals (e.g., [Anderson2002a, Anderson2002b, Alvarez2001, Borowsky1997]), but they considered only online redundancy techniques (mirroring and RAID 5). Our dependability design work builds on the experience gained there.

We also leverage existing work that deals with specifying and evaluating dependability requirements. Examples include a declarative method of specifying data performability and reliability requirements [Keeton2002, Wilkes2001], and ways to measure the availability of RAID systems [Brown2000].

Several recent papers from CNT (http://www.cnt.com) are particularly helpful in providing an overview of the growing regulatory and legal pressures on disaster tolerance in the financial industry. They provide some information about outage penalties, but little on the cost of lost data.

7 Conclusions

Data *must* be dependable: the cost of losing it, or even of losing access to it, is simply too high. The complexity of the systems to achieve this goal, and the range of failure cases they must tolerate, is increasing rapidly. What's more, human designers don't always make the right choices, or understand the ramifications of the choices they do make.

The solution is to automate the design of data dependability solutions. Our design tool provides numerous benefits over the manual approach employed today. By treating the question as an optimization problem, it can search the entire design space and find the best alternative for a set of business requirement inputs. Since the process is automated, it is easy (and fast) to evaluate a broad range of inputs to support "what if" analyses. Furthermore, the tool doesn't just provide "black box" answers, but can provide insights into the choices it makes, so that customers can understand and interpret its decisions.

Opportunities exist to extend this work in several different directions: improving the quality of the data protection technique models; enhancing the financial exposure model; extending the scope of the solutions and failures imposed; making it easier to add new data protection and recovery techniques to an existing solver; and generating easy-to-use visual interfaces to the tool itself. We hope to explore many of these avenues.

More details can be found in an extended version of this paper [Keeton2004].

Acknowledgements

Thanks to Minwen Ji and Alistair Veitch for helping us understand the intricacies of remote mirroring, Fereydoon Safai for helping with our test infrastructure, and Christos Karamanolis, Arif Merchant and our shepherd, Chandu Thekkath, for their helpful feedback on earlier versions of this paper.

References

[Alvarez2001] G. A. Alvarez, E. Borowsky, S. Go, T. H. Romer, R. Becker-Szendy, R. Golding, and J. Wilkes. *Minerva: an automated resource provisioning tool for large-scale storage systems.* ACM Transactions on Computer Systems **19**(4):483–518, November 2001.

[Anderson2002a] E. Anderson, M. Hobbs, K. Keeton, S. Spence, M. Uysal, and A. Veitch. *Hippodrome: running circles around storage administration.* Proc. File and Storage Technologies (FAST), pp. 175–188, January 2002.

[Anderson2002b] E. Anderson, R. Swaminathan, A. Veitch, G. Alvarez, and J. Wilkes. *Selecting RAID levels for disk arrays.* Proc. File and Storage Technologies (FAST), pp. 189–201, January 2002.

[Borowsky1997] E. Borowsky, R. Golding, A. Merchant, L. Schreier, E. Shriver, M. Spasojevic, and J. Wilkes. *Using attribute-managed storage to achieve QoS.* 5th International Workshop on Quality of Service (IWQoS'97), June 1997. Available from http://www.hpl.hp.com/SSP/papers.

[Brown2000] A. Brown and D. Patterson. *Towards availability benchmarks: a case study of software RAID systems.* Proc. 2000 USENIX Annual Technical Conference, pp. 263–276, June 2000.

[CNT2003] *Legal requirements for data retention and recovery in the financial industry*. Report PL731. Computer Network Technology Corporation. Sep. 2003. http://www.cnt.com/literature/documents/PL731.pdf, accessed Oct. 2003.

[CNT2003b] *Disk mirroring -- local or remote*. Report PL512, Computer Network Technology Corporation, August 2003. http://www.cnt.com/literature/documents/pl512.pdf, accessed Oct. 2003.

[Cougias2003] D. Cougias, E. L. Heiberger, K. Koop. *The Backup Book: Disaster Recovery from Desktop to Data Center*. Network Frontiers, LLC, 2003.

[daSilva1993] J. daSilva and O. Gudmundsson. *The Amanda network backup manager*. Proc. 7th USENIX Systems Administration Conference (LISA), pp. 171-182, Nov. 1993.

[Deavours2002] D. D. Deavours, et al. *The Möbius Framework and its implementation*. IEEE Transactions on Software Engineering, **28**(10):956–969, October 2002.

[EagleRock2001] *Online survey results: 2001 cost of downtime*. Eagle Rock Alliance Ltd, Aug. 2001. http://contingencyplanningresearch.com/2001 Survey.pdf accessed May 2003.

[EMC2004] EMC Networked Storage ROI & TCO Calculator. http://www.emc.com/forms/commercial/roi_calculator/project.jsp accessed January 2004.

[EMC–SRDF] *Symmetrix Remote Data Facility product description guide*. June 2000. EMC Corporation.

[GAMS1998] A. Brooke *et al*. *GAMS: a user's guide*. GAMS Development Corp.: Washington, DC. 1998.

[Haverkort2001] B. Haverkort, R. Marie, G. Rubino and K. Trivedi, eds. *Performability modeling: techniques and tools*, John Wiley & Sons, May 2001.

[HP–XP–CA] *hp continuous access xp.* Product brief 5980–8608EN, 2001. Hewlett-Packard Company.

[ILOG2002] *CPLEX 8.0 user's manual*. Ilog, Inc: Mountain View, CA. July 2002.

[Ji2003] Minwen Ji, Alistair Veitch, and John Wilkes. *Seneca: remote mirroring done write*. USENIX Technical Conference (USENIX'03) pp. 253–268. June 2003.

[Keeton2002] K. Keeton and J. Wilkes. *Automating data dependability.* Proc. 10th ACM-SIGOPS European Workshop, pp. 93–100, Sept. 2002 (Saint-Emilion, France).

[Keeton2004] K. Keeton, C. Santos, D. Beyer, J. Chase and J. Wilkes. *Designing for disasters – extended version*. Hewlett-Packard Laboratories Technical Report HPL-2004-16, available from http://www.hpl.hp.com/SSP/papers.

[Lee1996] Edward K. Lee and Chandramohan A. Thekkath. *Petal: distributed virtual disks*. Proceedings of ASPLOS VII. Published as *SIGPLAN Notices* **31**(9):84-92. Oct. 1996.

[Marcus2003] E. Marcus and H. Stern. *Blueprints for High Availability*. Wiley Publishing, Inc., 2003.

[Patterson2002] H. Patterson, S. Manley, M. Federwisch, D. Hitz, S. Kleiman, and S. Owara. *SnapMirror: file-system-based asynchronous mirroring for disaster recovery*. Proc. File and Storage Technologies (FAST), pp. 117–129, Jan. 2002.

[Preston1998] W. C. Preston. *Using Gigabit Ethernet to backup six terabytes.* Proc. 12th USENIX Systems Administration Conference (LISA), pp. 87-96, Dec. 1998.

[SEC2002] *Summary of "lessons learned'" from events of September 11 and implications for business continuity*. US Securities and Exchange Commission: Washington, DC. February 2002. http://www.sec.gov/divisions/marketreg/lessonslearned.htm

[Sun2004] Sun Microsystems Storage and Backup Requirements Calculator. http://www.sun.com/storage/tape/tape_lib_calculator.html, accessed January 2004.

[Wilkes2001] J. Wilkes. *Traveling to Rome: QoS specifications for automated storage system management.* Proc. the International Workshop on Quality of Service (IWQoS), pp. 75–91, June 2001.

[Williams1999] H. P. Williams. *Model building in mathematical programming*. John Wiley & Sons: New York. 4th edition, 1999.

[Wylie2001] J. Wylie, et al. *Selecting the right data distribution scheme for a survivable storage system,* Technical report CMU-CS-01-120, Carnegie Mellon University, May 2001.