From International Computer Performance and Dependability Symposium, Erlangen, Germany, April 1995, pp.285–294

MODELING RECYCLE: A CASE STUDY IN THE INDUSTRIAL USE OF MEASUREMENT AND MODELING

Luai M. Malhis[†‡], Stephen C. West[†], Latha A. Kant[‡], and William H. Sanders[*]
[†]IBM Corporation
Storage Systems Division
Tucson, AZ 85744
[‡] Dept. of Electrical and Computer Engineering
University of Arizona
Tucson, AZ 85721
[*] Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801

Abstract

Large-scale data storage systems rely on magnetic tape cartridges to store millions of data objects. As these tapes age, the resident data objects become invalid; consequently, less and less of the tape potential capacity is effectively utilized. To address this problem, data storage systems have a facility, called "recycle" in this paper, that transfers valid data objects from sparsely populated tapes onto new tapes, thus creating empty tapes for reuse. A high performance recycle process is needed to keep the number of tape cartridges to a minimum, and to maintain a continuous supply of empty tapes for storing newly created data objects. The performance of such processes is not easy to determine, and depends strongly on the data stored on the tapes, the speed and characteristics of the computer on which recycle is executed, and the nature of the algorithms themselves. This paper documents an extensive effort to evaluate a proposed recycle algorithm, using field workload data, laboratory measurements, and modeling. The results of the study were used to improve the recycle process, and were later verified through field trials. In addition yielding the results themselves, the effort illustrated that modeling and measurement, in an industrial setting, can indeed be used successfully in the design process.

1 Introduction

While system-level model-based performance evaluation is an active research area at many universities, its use in industry is not as widespread. In many cases, large-scale systems are designed in an *ad-hoc* manner, with validation (or disappointment regarding) system performance coming only after an implementation is made. This is particularly true in the area of software development, where algorithms are often devised and implemented without a clear understanding of how they will impact the performance of the system in which they are embedded. Extensive measurement and testing is often done after an implementation is complete, but is then too late to avoid design mistakes that lead to poor performance. This does not need to be the case. Modern modeling tools and techniques, coupled with measurements done on similar software, can yield accurate performance predictions that can be used in the design process.

We illustrate the effective industrial use of modeling, coupled with measurement, by reporting on the study of an algorithm to manage a large tape archive. The system (both hardware and software) considered was responsible for the "recycle" operation in the IBM's Data Facility Storage Management Subsystem Hierarchical Storage Manager (hereinafter called HSM). Recycle is the process of moving valid data objects from partially-filled tapes (called volumes) in a data archive to output volumes and releasing the previously partially filled volumes, now empty volumes, to a pool of scratch volumes for subsequent reuse. The recycle process prevents the unbounded growth in the number of volumes in the archive, many of which may be nearly empty, and contributes to the efficient utilization of the potential archive capacity.

The activity reported herein was completed over a six-month period at IBM and at the University of Arizona. During this time, a "project team" at IBM, whose task was to revise the algorithms used in the recycle operation, interacted with a "modeling and

measurement team," composed of IBM employees and University of Arizona researchers. The task of the modeling and measurement team was to build accurate models of both existing and proposed recycle algorithms, and to aid in the development of more efficient recycle algorithms. The modeling team built two models: one of the existing recycle algorithm, and one that contained many proposed changes. We obtained parameter values for these models from actual trace data collected in the laboratory, and from existing recycle customer sites while executing the existing recycle algorithm, and also from a careful study of several real-life workloads (tape archives) from customer sites.

Comparing the results of both models under the same workloads gave the project team a quantitative measure of changes of performance obtained by redesigning the system. Such measures could only come from modeling, because the proposed modifications had not yet been implemented. These results helped to guide the development effort of the project team by focusing its attention on items that would have the largest gain in performance. Furthermore, we validated the results of the model by field measurements on the revised recycle software.

We expressed the existing and proposed recycle algorithms as composed stochastic activity networks (SANs) [1], and solved using the terminating simulator in the *UltraSAN* [2] modeling environment. For more detailed discussion of SANs and *UltraSAN* see [1, 2, 3]. The results are significant because they show that modeling, coupled with measurement, can be effectively used in the industrial software design process to predict the performance of alternative algorithms and software implementations. Furthermore, they show that SANs are indeed a reasonable method to express specific software algorithms, and *UltraSAN* can be used to efficiently solve large, industrial, models.

The remainder of the paper is organized as follows: Section 2 provides an overview of the existing system and the proposed modifications, Section 3 discusses the measurement activity that provided guidance to build the models, Section 4 discusses the models, to help the reader understand how complex software can be represented as SANs, Section 5 discusses the results obtained from the models, and concluding remarks are presented in Section 6.

2 Background and Problem Definition

HSM provides facilities for managing data sets (files) on storage devices. HSM also migrates primary data between storage devices (off-loads of infrequently accessed data sets to slower media) and creates backup copies of unmigrated primary data (creates redundant copies of data sets without moving the primary data sets). The HSM storage administrator controls the migrate and the backup processes to achieve the migrate and backup rates required by the user's installation (customer site).

Over time, these backup and migrate actions result in storing large amounts of data on magnetic tape cartridges (called volumes). As volumes age, the resident data sets (also called "objects") slowly become invalid for the following reasons: 1) when data sets change frequently, they cause multiple versions of the same data set to be backed up. When the number of copies (versions) exceed some defined limit, the oldest copy becomes invalid, 2) when the primary data set no longer exists, the backup versions become invalid, 3) when a migrated data set is recalled to primary storage from a tape volume, the data set on the volume becomes invalid, and 4) when the age of a data set exceeds some predefined value, the data set becomes invalid.

Because of these invalidations, tape volumes (which may each contain tens to thousands of data sets) contain increasingly large amounts of invalid data. Consequently, the potential capacity of tape volumes in an archive is less utilized. To address this inefficiency, HSM has an operation called "recycle." The recycle operation selects valid data sets on sparsely populated volumes to be transferred onto another volume, thus aggregating valid data sets from many volumes onto a single volume. Recycle then frees the newly emptied volumes for reuse. For example, if 100 volumes with an average of 10% valid data are recycled, the process recovers these 100 volumes at the expense of 10 new full volumes, or a net tape gain of 90 volumes.

The original implementation of recycle has been distributed widely with thousands of customer installations. Recently, it became more common and necessary for customers to run their operations around the clock, leaving little idle capacity for needed, but time-consuming, operations such as recycle. To make this possible, the recycle project team within IBM was asked to propose modifications to the recycle algorithm to increase its efficiency (increasing the rate of tape gain during the execution of recycle) and, in turn, to reduce the number of tapes necessary to store a given amount of valid data.

In particular, the following changes were proposed to the existing Recycle algorithm:

Multi-tasking The original recycle implementation uses a single task to transfer valid data sets from a single input volume to a single output volume. The project team proposed that multiple, but independent, input-output pairs be supported. The intent was to obtain an increase by a factor equal to the increased number of input-output pairs.

List Building and Sorting The original recycle implementation processes the meta-data catalog in a specific collating order. Each new invocation of recycle starts the new recycle search at the beginning

of this collating order. In particular, volumes are selected by searching the catalog, looking for volumes that meet the recycle criteria (the fraction of valid data on the volume is less than some value). When a volume that meets the recycle criteria is encountered, the search operation is suspended and the volume is immediately recycled. After the volume is processed, the search is resumed. Often, customers terminate the recycle process before all the volumes that meet the recycle criteria are processed, for example, when a given number of empty tapes are produced. This selection and processing algorithm causes the following: 1) many empty or nearly empty volumes not to be reclaimed (because of early termination) and, 2) volumes with a higher percent valid to be processed before volumes with a lower percent valid.

Such behavior does not provide the greatest rate of tape gain over a fixed execution period. The project team thus proposed that the catalog records be searched at the beginning of the recycle operation, and volumes that meet the recycle criteria be processed in increasing order of percent valid data. Such an implementation should produce the maximum tape gain rate over any period of execution of the recycle process, if the overhead incurred in processing the list were small relative to the time to recycle volumes.

Immediate Queue If the overhead in building the sorted list is not small, processing of volumes can be delayed significantly, which can cause a customer to perceive an inefficiency in the recycle operation. Because the overhead was unknown (prior to the modeling project), the project team proposed maintaining a second queue, which was not sorted in increasing percent valid order, but met the recycle criteria. The project team proposed an algorithm that can place certain volumes on the second queue depending on their percent valid and the current length of queue. The volumes on the second queue can then be recycled while the list is being built and sorted, avoiding the delay in the start of processing.

Input Drive Allocation The project team also recommended changing the way input drives were allocated in recycle. In particular, the original implementation requested a tape drive allocation for each input volume processed, and released the allocation following the recycle of the volume. The repeated allocations were made to allow competition for the tape drive resources and to avoid blocking other activities while the operator retrieved the next volume. With the advent of automatic tape libraries (hence reducing the time necessary to change tapes), it was deemed appropriate to consider retaining the input drive allocation over multiple input tapes.

Multiple Read/Write Buffers The original implementation uses a single fixed size memory block to buffer data from the input tape to the output tape. This implementation required that tape reads and writes be done in a sequential fashion, with a write to the buffer beginning only after the previous read completed. It was hypothesized that this serialization contributed significantly to the inefficiency of the original implementation. Consequently, the project team proposed that multiple read/write buffers be used, in a manner that allowed parallel reads and writes to occur.

Connected Set Processing Finally, the project team recommended changing the way volumes were selected for processing, considering the basic unit of selection as a "connected set," rather than an individual volume. Connected sets are formed when a single data set spans more than one volume. To achieve maximum fullness on each migrate or backup volume, a new data set is started on a volume if the expected remaining capacity of the volume is more than some value. If the started data set is not completely written when a maximum allowed capacity is reached, the output volume is demounted and a new empty volume is mounted in the target drive. The suspended data set is continued on the new volume, which causes a single data set to span two volumes (or more, for very large data sets).

The data set that exists on more than a single volume is called a *spanned data set.* The volume set (of size one or more) with no spanned first (valid) data set, no spanned last (valid) data set, and with zero or more (valid) spanned data sets form a single *connected set.* Note that a connected set can be of size one. The original implementation based recycle selection upon an individual volume's percent valid calculation. When a selected recycle volume had one or more spanned data sets, the volumes that contained part of the spanned data sets were mounted and the partial data set was recovered. The other data sets on these incidentally selected volumes were not recycled unless the volume was selected upon its own merits later in the recycle sequence. The new method proposed was to consider each connected set as a single unit for recycle and the total connected percent valid value to be the criterion for selection.

It was thought that these changes to Recycle would increase its performance, but it was not clear the extent to which the performance would improve, or which of the proposed design changes would have the most significant impact. To investigate these options, we built a detailed model of the recycle operation, using real customer workload data (measured in the field), and software execution time measurements, collected in the laboratory and at real customer sites. We describe these workload and software measurements in

the next section.

3 Source of Data and Workload

When evaluating real systems, either existing or to be implemented, determination of reasonable values for model parameters is generally the most difficult and time-consuming activity in the modeling process. Whereas a model of an academic nature may allow simplifying assumptions and a restricted focus, real systems tend to have less than ideal implementations and to require models that can address a broad spectrum of questions. It is thus important to ensure that these real-life imperfections, perturbations, and broad scopes are addressed in their models.

With an accurate understanding of the system to be modeled, it is reasonably straightforward to construct models that represent all of the necessary system steps in the proper sequence, and to make all of the correct choices for the algorithms being modeled. To also provide accurate and meaningful results, the model must be as follows: 1) constructed with the correct activity times (service, delay, wait, and so on) for all of the resources implemented in the model, and 2) exercised with a meaningful workload.

The modeling and measurement team developed a basic understanding of the existing system and the intended enhancements through extensive discussions with the project team. Though these discussions were informative and important, much more detailed and more precise timing information than was known by the developers was required to build accurate models. This detailed information was obtained from three primary sources: 1) trace data from existing user installations, 2) trace data from special test cases run in IBM's performance measurement laboratory, and 3) Control Data Set (CDS) information from four existing customer installations (a customer's CDS information specifies, precisely, the type of information stored on his tape archive).

The data that was collected was extensive, and space does not permit a detailed accounting of all measurements and analyses in this paper. Instead, we will try to highlight the types of information that went into the developed model to give the reader an indication of type, and level of detail, of information that was included.

The actual sequence of software events encountered during the execution of the recycle algorithm was determined by analyzing trace data from an existing user installation. This analysis produced both an understanding as to which steps in the sequence were significant and, in many cases, the actual timing information that was used in the model. The key sequence and timing information included: 1) volume selection sequence and times, 2) valid data set search, selection, and processing sequence and times, and 3) tape activity (waits, mounts, demounts) times.

One important issue that we needed to address was

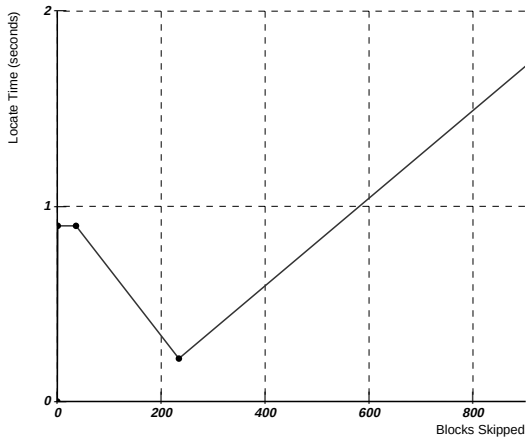

Figure 1: Tape locate time as a function of invalid blocks skipped

the timing associated with the physical movement of tape within a volume. Though the user installation trace data allowed us to determine tape motion delays encountered while moving over invalid data sets, we had no way of determining how much media (that is, data blocks) was skipped. To determine these skip times to locate to the next valid data set, we conducted laboratory measurements, using the original recycle implementation, that allowed us to measure the locate time as a function of the number of invalid blocks skipped. The derived relationship, shown in Figure 1, has three distinct regions of operation: 1) The initial region, which is dominated by the tape repositioning time, has little to do with the number of blocks being skipped; 2) The middle region, where the locate time actually decreases for a period of time as a result of reduced tape repositioning; 3) The final region, which has a linearly increasing time to locate to the next valid data set.

The tests of recycle conducted in the laboratory also helped to refine some of the numbers that we had previously obtained from the user installation trace data, for example, the precise read/write buffer timings and a more precise split of what was CPU utilization time and IO or disk access time, for many of the CDS access steps.

To assure the validity and acceptance of the model results, we developed a realistic workload for the models from the user installations archive meta-data records. These records were in the CDSs. We had access to these single point-in-time CDSs from four user installations. The CDSs contain a complete set of records relating to the state of all currently valid tape data sets. These records contain data set names, dates, sizes, location, and so on. The CDS information was received in its raw, unprocessed form and custom data reduction programs were written to extract the information deemed to be interesting. We were able to determine a number of important distri-

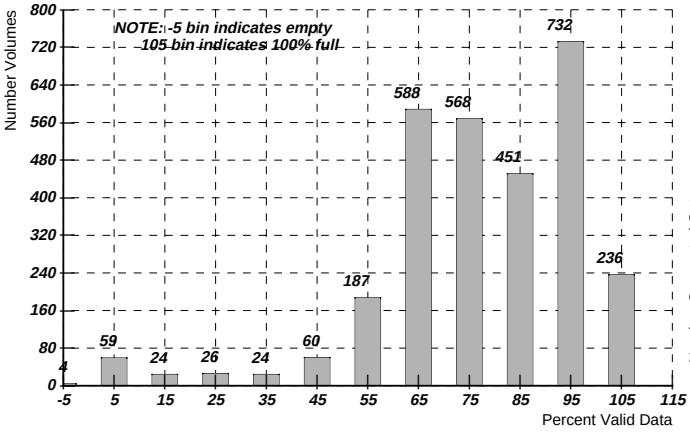

Figure 2: Percent valid distribution for volumes

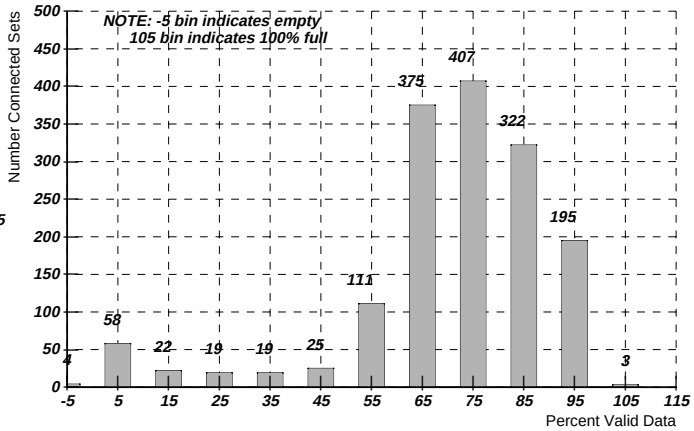

Figure 3: Percent valid distribution for connected sets

butions for both migrate and backup type data sets. In particular, we derived the following: 1) fraction of volumes containing particular amounts of valid data (10 percent resolution), 2) distribution of number of connected sets by percent valid bin, 3) distribution of number of volumes per connected set by percent valid bin, 4) probability of 0, 1, and 2 spanned data sets (per volume) by percent valid bin, 5) distribution of non-spanned data set size by percent valid bin, 6) distribution of spanned data set size by percent valid bin. All conditional distributions (conditioned upon the percent valid values) were determined for each 10 percent bin of the percent valid distributions. In addition to the distributions above, we determined the fraction of volumes that are empty among all volumes in the record.

Figures 2, 3 and 4, are examples of the type of information that we extracted from the customer installation CDSs (these are from an installation that we chose to use as a workload). Figure 2 shows the distribution of migrate volumes by percent valid bins.

Figure 3 shows the distribution of migrate connected sets by percent valid bins. As was typical in all of the installations, the ratio of volumes to connected sets was closest to 1 at low percent valid and grew progressively larger as percent valid became larger. The conditional connected set size distributions were determined, by percent valid bin, and utilized in the model. For all volumes and connected sets, the ratio was 2959/1566 or 1.9 for this installation.

Figure 4 shows the size (number of volumes per connected set) distribution for the connected sets. Most connected sets are small in size, 1 or 2 volumes, but there are some of size 10, 15, and 20 volumes. These larger connected sets are always the newest connected sets where the data sets have not started to become invalid, very close to 100% valid.

The data capture phase consumed from 40% to 50% of the total work of the modeling and measurement team. In retrospect, the substantial time spent on

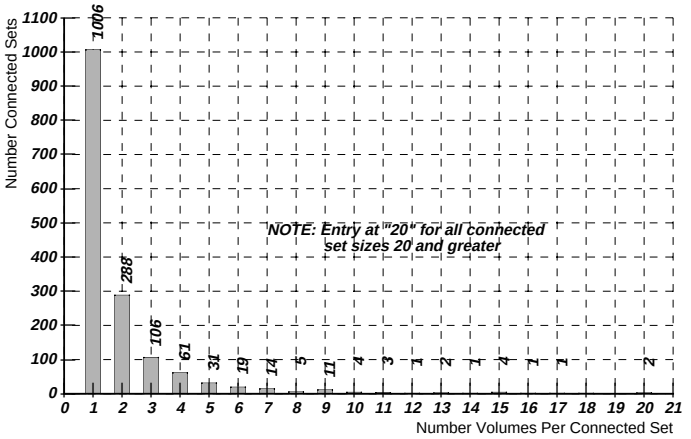

Figure 4: Size distribution for connected sets

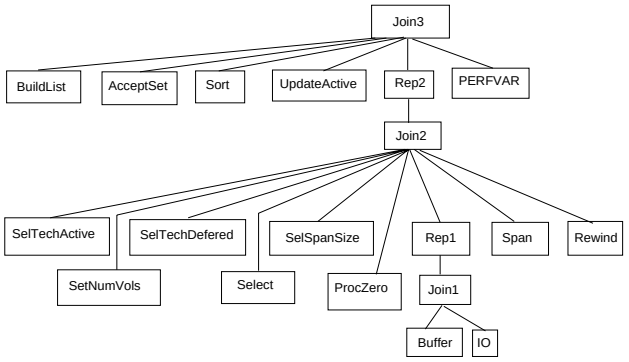

Figure 5: Composed model for Stage 1 recycle operation

measurement and data analysis paid off, because it made the construction of models much easier. This activity extracted data from a number of different sources, with customer installation supplied data and local laboratory measurement data being the most significant source of data. In the next section, we discuss the models, and how they use the data discussed we presented.

4 SAN Models for Existing and Proposed Recycle Algorithms

We built two SAN models, one for the original algorithm and one for the proposed modification. We maintained as much commonality between the two models as possible, introducing differences only as necessary to implement the proposed changes in the algorithms. Because of space limitations, we are unable to discuss in detail both of the models. Instead, we describe the structure of the SAN model of the proposed algorithm, and, in some detail, a small part of the model of the proposed algorithm that implements the list building operation. In this way, we hope you will gain an appreciation of the complexity of the models, and how we integrated the algorithms and measured workload data into the SAN models. For a more detailed description of both SAN models, see [4].

Figure 5 shows the "composed model" for the proposed recycle algorithm. A composed model [5] is built from several SAN submodels using "replicate" and "join" operations. These operations permit the construction of hierarchical (or composed) models from existing SAN models. The *replicate* operation replicates a subnet a certain number of times. The replicate operation thus allows the construction of composed models that consist of several identical component subnets. The combination of several different subnets is accomplished using the *join* operation. The join operation produces a composed model that is a combination of the individual subnets.

Each subnet in the composed model in Figure 5 represents some of the processing steps in the proposed recycle algorithm. For example, the subnet labeled **buildList** represents the process of searching through the meta-data records and determining which connected sets meet the recycle criterion. The subnet labeled **acceptSet** determines whether the selected connected set for recycle should be inserted into the immediate queue, to be recycled immediately, or inserted into the defered queue, to be recycled after the completion of the list building and sorting process. Sorting the list is represented by the subnet **Sort**.

The subnets representing a single task are joined together using node **Join2**. These joined subnets are then replicated a number of times (equal the number of tasks in the system), using node **Rep2**, to represent the multi-tasking scenario. The multiple read/write buffers for each task are represented by the two subnets **buffer** and **io**, and the two nodes **Join1** and **Rep1**. The number of times **Rep1** is replicated reflects the number of read/write buffers available to each task to transfer the data blocks. Finally, the node **Join3** joins all the subnets needed to build the recycle model for the proposed algorithm.

Each leaf in the composed model is a *stochastic activity network* (SAN) [1]. Figure 6 shows an example SAN from the composed model shown in Figure 5. Stochastic activity networks are a stochastic extension to Petri nets. Structurally, they consist of *activities, places, input gates,* and *output gates.* Activities (*startSel* and *contBuild* in Figure 6) represent activities of the modeled system whose durations impact the ability of the system to perform. *Places* (*selNext* and *exit1* in Figure 6) are used to represent the "state" of a system and may contain *tokens*. *Cases* associated with activities (represented as small circles on one side of an activity, *repeat* in Figure 6) permit the realization of uncertainty with probabilistic choices concerning what happens when an activity completes. *Input gates* and *output gates* permit flexibility in defining enabling and completion rules. In this model, we use places, along with input and output gates, to control the processing flow of the recycle operation. We use activities without cases to represent the various timings in the recycle process and we use activities with cases to represent the various distributions (described in the previous section) that define the model workload.

To illustrate the use of SANs to build the recycle model, we describe in some detail the structure and the function of the subnet labeled **buildList** shown in Figure 6. Specifically, for each connected set, we must probabilistically determine a percent valid value from the density function shown in Figure 3. This determination can be accomplished by using a single *activity* with a number of cases equal to the number of possible percent valid values a connected set can have. If the percent valid values must be specified to within one percent resolution, an *activity* with 100 cases is needed. On the other hand, if the percent valid resolution is required to within 0.1 percent then an *activity* with 1000 cases (or some combination of several *activ-*

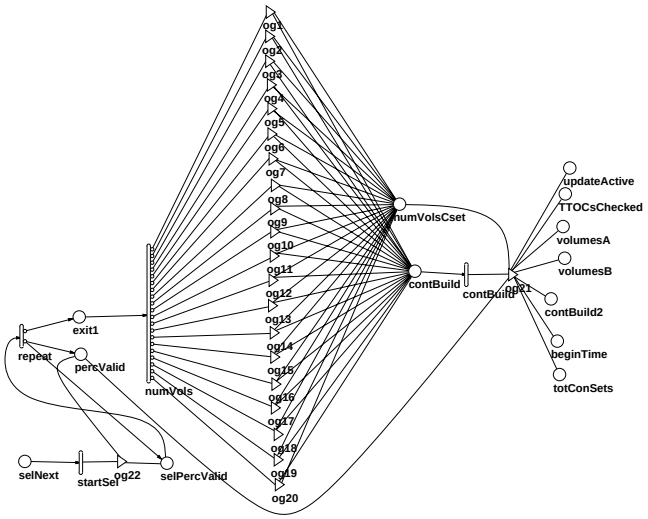

Figure 6: SAN Model: *buildList*

ities each with many cases) is required. This approach is impractical, especially if some flexibility is needed to decide at what resolution of the percent valid values to run the model. A more practical approach is to use conditional distribution for each percent valid value. We use the conditional distribution to compute the probability that a connected set have a percent valid value equals x, assuming that the percent valid value for that connected set is greater than or equal to x.

We modeled the percent valid value selection process using a single activity with two cases (activity *repeat* in Figure 6). Each time this activity is enabled, we compute the probability distribution function of the density function shown in Figure 3. Then two probabilities for the two cases associated with this activity. Let the variable called **accept_prob** be the probability value associated with case 1, and the variable called **reject_prob** be the probability value associated with case 2. If the model is to run with percent valid resolution equals 0.1 percent, there are 1001 different percent valid values (ranges from 0.0 percent to 100.0 percent) that a connected set can have. The place *percValid* contains a number of tokens that represents one of the 1001 possible values. For example, if the number of tokens equals 125, the percent valid value represented is 12.5 percent. The number of tokens in this place are set to zero before a percent valid value is selected for a connected set using output gate *og22* in Figure 6.

The computed PDF, conditioned on the number of tokens in place *percValid*, uses the following procedure to compute **accept_prob** and **reject_prob** values. The variable **num_tokens** represents the number of tokens in place *percValid*. If **num_tokens** equals zero, then **accept_prob** is set to the value of the PDF at zero, and the **reject_prob** is set to 1.0 - **accept_prob**. If **num_tokens** is not zero but less than 1000, the two variables **upper** and **lower** are assigned the values **num_tokens** + 0.5, and **num_tokens** - 0.5, respectively. Also, the value of the PDF at the point **lower** is **PDF_lower** and the value of the PDF at point **upper** is **PDF_upper**, and variable **total_prob** is 1.0 - **PDF_lower**. Then, **accept_prob** is set to (**PDF_upper** - **PDF_lower**)/**total_prob**, and the **reject_prob** is set to 1.0 - **accept_prob**. Finally, if number of tokens in place *percValid* equals 1000, then **accept_prob** is set 1.0 and **reject_prob** is set to 0.0. Thus, conditioned on the number of tokens in place *percValid*, the PDF is used to determine the probability of accepting or rejecting that percent valid value for the connected set.

When activity *repeat* completes, either case 1 or case 2 is chosen. If case 1 is chosen, a token is added to place *exit1*. In this case, the connected set is assigned the percent valid value represented by the number of tokens in place *percValid*. If case 2 is chosen, then that percent valid value is rejected. In this case a token is added to place *percValid* and place *selPercValid*. Hence, the process of accepting or rejecting the next possible percent valid value is repeated. This approach gives greater flexibility in running the model at any specified resolution for percent valid.

Moreover, we use the activity *numVols*, and the cases associated with this activity to represent the conditional distributions (conditional upon the percent valid values) of the number of volumes in a connected set. The unconditional distribution of the number of volumes in a connected set by percent valid bins is shown in Figure 4. From this distribution, the possible number of volumes in a connected set can vary from one to twenty. Hence, an activity with 20 cases can be used to model 20 different possibilities.

In some activities, the time to do an operation is not a single value but rather a function of some value. For example, the time required to locate the next valid data set to be transferred from the input tape is a function of the number of invalid blocks that must be skipped to move to the next valid data set. This locate function is shown in Figure 1. In this case, the activity completion time in the SAN model must be representative of such functions. The function specifying the activity completion time of the locate operation is shown, in Table 1, example of the possible complexity of such functions.

When completed, the model of the proposed recycle algorithm consisted of 16 subnets, 162 input and output gates, 177 places and 77 activities. The timing information and the workload were then inserted into the models, and the models were simulated using the terminating simulator in *UltraSAN*. Confidence intervals are generated by the simulator, and replications are done until a desired level of confidence is reached.

Table 1: Activity time distributions for SAN *select*

Act.	*Dist.*	*Parameter values*
skip	det.	
	value	*/* skip time is based on number of blocks */* *if (blkstoskip == 0)* *return(0.0);* *if (blkstoskip < 37)* *return(0.9007615);* *if (blkstoskip < 236)* *return(1.029212 − 0.003453 * blkstoskip);* *else* *return(−0.239565 + 0.001940 * blkstoskip);*

5 Results

We present results here for a single workload for a single, sample migrate account, although many workloads were studied in the course of our work. The number of migrate volumes managed was 6000. To illustrate the type of output available, we examine two parameters of interest in this section. Subsection A, titled **Net Tape Gain**, contains results on the rate at which tape volumes are recovered. Subsection B, titled **Multiple Buffers**, discusses the effect of multiple buffers on the Net Tape Gain.

5.1 Net Tape Gain

The measure of most interest to the development team and the user community is at what rate are tapes released for reuse, or in other terms, the number of tapes freed for new use in some period of time.

The performance variable net tape gain (NTG), is the number of tapes released during the recycle process. If we let `empties` represent the empty candidate volumes, `freed` the number of non-empty candidate volumes and `filled` the number of output tapes used to hold all of the valid data in the non-empty candidate volumes, then NTG = `freed` + `empties` - `filled`.

Net Tape Gain: Stage 0 Figure 7 gives the cumulative NTG (in volumes) as a function of recycle time (in hours) for five different preset threshold values for Stage 0.

The curves in Figure 7 illustrate the following: First, the net tape gain for a given threshold value increases linearly with recycle time until all the 6000 input tapes have been processed, after which it flattens out because there are no additional tapes to recycle. For example, the curve for 30 percent threshold

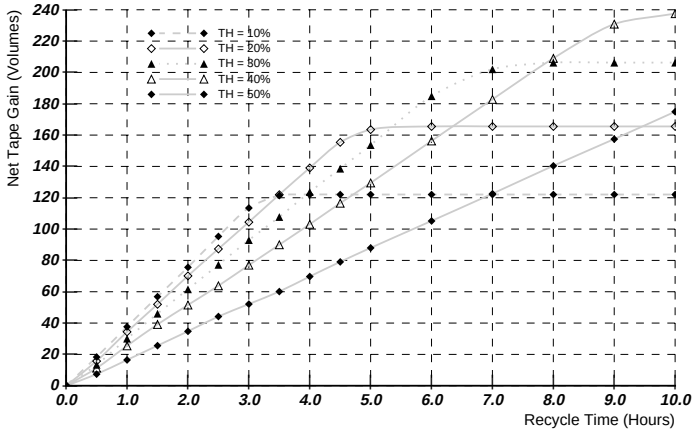

Figure 7: Net Tape Gain (Stage 0)

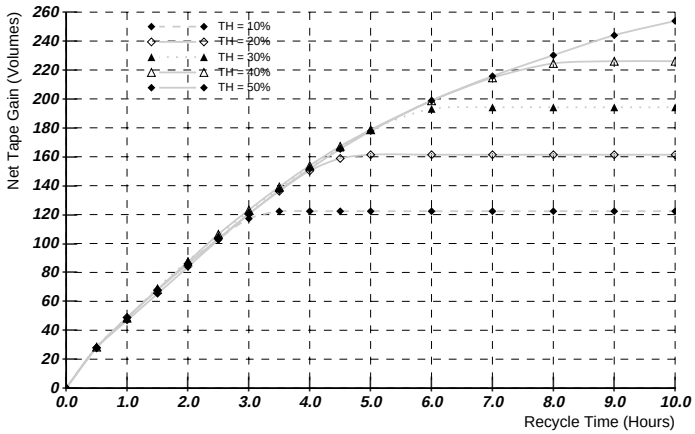

Figure 8: Net Tape Gain (Stage 1)

rises linearly for apprximately seven hours, which represents the time to scan the 6000 input volumes and to recycle those cartridges with less than or equal to 30 percent valid data. Each plateau implies that the 6000 volumes were scanned and there are no more tapes to be reclaimed. Second, the number of tapes reclaimed per hour is inversely proportional to the threshold value because larger threshold values have a greater amount of valid data to be transferred. This implies that a greater amount of time needs to be spent in the data transfer phase, which reflects in a smaller number of tapes released per hour. Third, the recycle time increases with the threshold value, with this time varying from three and a half hours to nine hours for threshold values of 10 percent through 40 percent. Because the viewing window was restricted to 10 hours, we can not determine the completion time for the 50 percent curve.

Net Tape Gain: Stage 1 The set of curves in Figure 8 give NTG as a function of time for Stage 1. Again, the time to recycle the 6000 volumes increases

as the percentage of threshold value increases, and the plateauing of the NTGs occur upon completion of the recycle process.

An important difference between Stages 0 and 1 is in the rate of NTG for different threshold values. While the rate of NTG differed between the various threshold values in Stage 0, Stage 1 exhibits little difference as the threshold value changes. This is due to the sorting process used in Stage 1, which always places the same volumes in the same order regardless of the threshold. The only effect the larger threshold values have is that they allow the recycle process to process further into the sorted list.

Thus, these curves are approximately the same and are within their simulation confidence intervals except for their completion points. For example, after the first 3 hours of recycle for the migrate account studied, the NTG in Figure 8 begins to flatten out for the 10 percent threshold value (indicating its completion). The curves for higher threshold values remain clustered until the curve for 20 percent threshold breaks away around the four and a half hour time period (marking its completion). This process continues until the maximum threshold is reached.

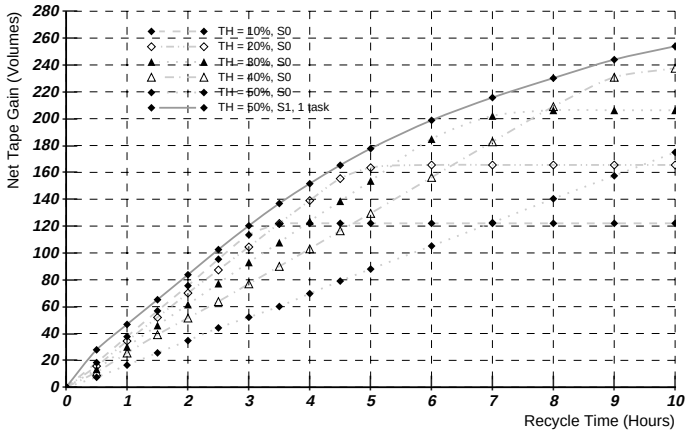

Figure 9: Net Tape Gain (Stages 0 and 1 with single task)

Comparisons between Stages 0 and 1 Figures 9 and 10 illustrate how the two recycle approaches compare. Because Stage 1 allows for the presence of multiple tasks and Stage 0 does not, Figure 9 compares both stages with 1 task, while Figure 10 compares Stage 0 with 1 task and Stage 1 with multiple tasks for a fixed threshold. We chose a threshold value of 50 percent for Stage 1 because it was consistent with the largest Stage 0 threshold and would provide the best comparison over the total threshold range.

Comparing the curves in Figure 9, we see that Stage 1 outperforms Stage 0, with the NTG of the former being higher than that of the latter for the same migrate account studied. The performance difference tends to become enhanced with increasing threshold values.

Figure 10 compares the NTG for Stages 0 and 1, with Stage 1 having multiple tasks. This figure shows the very significant gain obtained by increasing the number of tasks. Also note that the time to obtain a given level of NTG varies linearly across the number of tasks. For example, the time taken to recover 200 net tapes in Stage 1 is approximately six, three and two hours with one, two, and three tasks, respectively.

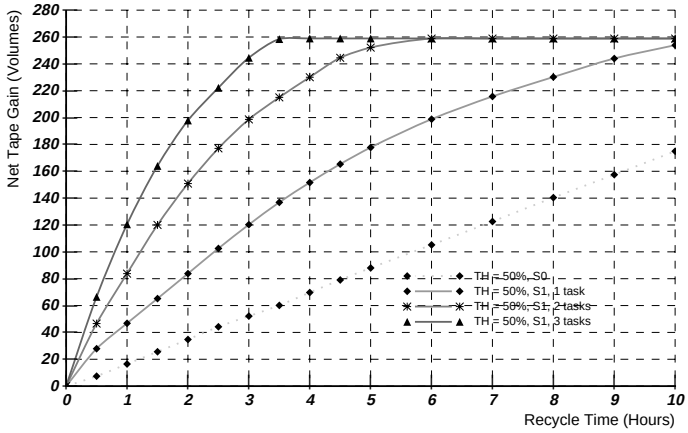

Figure 10: Net Tape Gain (Stages 0 and 1, 50% threshold, with multiple tasks)

5.2 Multiple Buffers

We show the net tape gain as a function of the number of buffers allocated to a single task in Figure 11. In this study, the recycle threshold is set to 50 percent. the number of volumes is set to 6000, and a single task processes tapes. During the first few hours ($\leq$ four hours), the processed connected sets have a very low percentage of valid blocks. If a connected set has a very small percentage of valid blocks, the amount

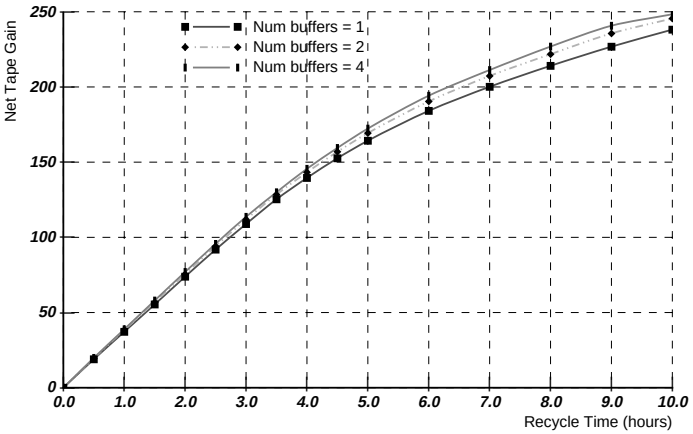

Figure 11: Changing number of buffers

of time spent in moving valid data from the source volume to the target volume is very small compared to total processing time for that connected set. Most of the total processing time is spent verifying whether a data set is valid or not, mounting new volumes into the input drive, locating to the next valid data set, and rewinding volumes. Therefore, changing the number of buffers from 1 to 2 to 4 has a very small impact on the net tape gain.

However, if the connected sets have a large percentage of valid blocks, the number of buffers used has some effect on net tape gain. The reading and writing of buffers are independent operations because they operate on independent drives. Therefore, increasing number of the buffers from 1 to 2 increases the net tape gain slightly because while one buffer is being read from the input drive, another buffer can be written to the output drive.

But, the increase in net tape gain is small because, regardless of what percentage of valid blocks a connected set has, the data movement time is still small compared to other processing times per input volume. Furthermore, the change in net tape gain when the number of buffers is increased from 2 to 4 is much smaller than in the case when the number of buffers is changed from 1 to 2. This minimal improvement is due to very little use of any serial resource.

To summarize, the multiple buffer concept has a small and variable impact on the recycle process. The greater the amount of valid data on the volume, the greater the enhancement. As the tape technology density increases, the usefulness of multiple buffers should increase.

The retention of the input drive allocation did not prove to be a significant enhancement in these models because we did not model contention for the drives from other applications. In a real environment with moderate to heavy drive utilizations, this approach may have noticeable performance gains for the recycle process.

Observations of system resources, such as CPUs and ATL Picker, utilization showed that there are periods during the recycle process when the host CPU, the serial IO, and the ATL picker utilizations peak to appreciable levels. These system resources may not be available at such levels and the recycle process may be adversely affected. The use of these resources has a linear increase as a function of the number of tasks in the system. The peak CPU utilization, serial I/O utilization, and picker utilization per input task are 2.7 percent, 3.1 percent, and 9 percent, respectively.

6 Conclusion

The results presented here served two purposes. First, they provided useful (and sometimes surprising) information to the project team. This information was used, wherever possible, in the current new release of recycle, and will be used in future releases. Second, the results illustrated that modeling and measurement can be used, very successfully, in an industrial setting. This realization is important, since modeling had not been used extensively in the past, due to uncertainty of its benefits for such software systems. Stochastic activity networks were shown to be an appropriate representation for the types of systems we are interested in, and made it easy to integrate measurement information collected in the field and laboratory. In addition, the work provided the necessary evidence of the utility of our approach, and resulted in the initiation of several new modeling projects, using stochastic activity networks and *UltraSAN*.

Acknowledgment The modeling team thanks Cindy Gallo, Jerry Pence, Tony Pearson, Lyn Ashton, and Don Lawver, from the IBM Storage Systems Division in Tucson, Arizona, and Bhavan Shah, originally a member of the university modeling team, now an employee of Intel in Portland, Oregon, for their support and contributions throughout the recycle modeling effort.

References

[1] Meyer, J. F., A. Movaghar, and W. H. Sanders, "Stochastic activity networks: Structure, behavior, and application", in *Proc. International Workshop on Timed Petri Nets*, pp. 106–115, Torino, Italy, July 1985.

[2] Couvillion, J., R. Freire, R. Johnson, W. D. Obal II, M. A. Qureshi, M. Rai, W. H. Sanders, and J. E. Tvedt, "Performability modeling with *UltraSAN*," *IEEE Software*, vol. 8, no. 5, pp. 69-80, September 1991.

[3] Sanders, W. H. and R. S. Freire, "Efficient Simulation of Hierarchical Stochastic Activity Network Models," in *Discrete Event Dynamic Systems: Theory and Applications*, vol. 3, no. 2/3, pp. 271–300, July 1993.

[4] Kant, L., L. M. Malhis, W. H. Sanders, B. P. Shah, and S. C. West, Reference Manual: *UltraSAN* models

for recycle modeling project, PMRL Technical Report 93-18, Department of Electrical and Computer Engineering, University of Arizona, December 1993.

[5] Sanders, W. H. and J. F. Meyer, "Reduced base model construction methods for stochastic activity networks," in *IEEE Journal on Selected Areas in Communications,* special issue on *Computer-Aided Modeling, Analysis, and Design of Communication Networks*, vol. 9, no. 1, pp. 25-36, January 1991.

[6] Sanders, W. H., *Construction and solution of performability models based on stochastic activity networks*, Ph.D. thesis, University of Michigan, Michigan, 1988.

[7] Performability Modeling Research Laboratory, *UltraSAN User's Manual*, Department of Electrical and Computer Engineering, University of Arizona, August 9, 1993.