ViPEr-HiSS: A Case for Storage Design Tools

Leana Golubchik   Joseph Dunnick   Jeffrey K. Hollingsworth

Department of Computer Science
University of Maryland
College Park, MD 20742
{leana,jow,hollings}@cs.umd.edu

Abstract

The viability of large-scale multimedia applications, depends on the performance of storage systems. Providing cost-effective access to vast amounts of video, image, audio, and text data, requires (a) proper configuration of storage hierarchies as well as (b) efficient resource management techniques at all levels of the storage hierarchy. The resulting complexities of the hardware/software co-design in turn contribute to difficulties in making accurate predictions about performance, scalability, and cost-effectiveness of a storage system. Moreover, poor decisions at design time can be costly and problematic to correct in later stages of development. Hence, measurement of systems *after* they have been developed is not a desirable approach to predicting their performance. What is needed is the ability to evaluate the system's *design* while there are still opportunities to make corrections to fundamental design flaws. In this paper we describe the framework of ViPEr-HiSS, a tool which facilitates design, development, and subsequent performance evaluation of *designs* of *multimedia* storage hierarchies by providing mechanisms for relatively easy experimentation with (a) system configurations as well as (b) application- and media-aware resource management techniques.

1 Introduction

The viability of large-scale multimedia applications, such as digital libraries, tele-medicine, digital special effects, and distance learning, depends on the performance of storage systems. Common among these applications is their high bandwidth and storage requirements, coupled with some form of real-time or continuity constraints. Such requirements pose a significant challenge to the design of hierarchical networked storage systems. Many companies, such as HP, IBM, and Intel, have identified the design and maintenance of large-scale storage systems as an important growth area, emphasizing the need for effective designs of storage systems.

Providing cost-effective access to vast amounts of video, image, audio, and text data, requires (a) proper configuration of storage hierarchies as well as (b) efficient resource management techniques at all levels of the storage hierarchy, including data placement, scheduling, prefetching, and caching. The resulting complexities of the hardware/software co-design in turn contribute to difficulties in making accurate predictions about performance, scalability, and cost-effectiveness of a storage system. Moreover, poor decisions at design time can be costly and problematic to correct in later stages of development. Hence, measurement of systems *after* they have been developed is not a desirable approach to predicting their performance. What is needed is the ability to evaluate the system's *design* while there are still opportunities to make corrections to fundamental design flaws. Consequently, design, analysis, evaluation, and development of hierarchical multimedia storage systems is no small undertaking, for which few have all the necessary resources.

In this paper we describe the design of ViPEr-HiSS, a tool which facilitates design, development, and subsequent performance evaluation of *designs* of *multimedia* storage hierarchies by providing mechanisms for relatively easy experimentation with (a) system configurations as well as (b) application- and media-aware resource management techniques *without* the need for extensive code development or access to *physical* storage devices. Specifically, we allow the design engineer to describe a system's design (including hardware configuration and corresponding resource management techniques) in terms of a combinations of physical, simulated/emulated, and analytically modeled storage devices.

We believe that work on the design and development of such tools represents a fundamentally important step in the design and performance evaluation of next generation hierarchical storage systems. Many systems today are moving towards applications whose primary function is to transfer large volumes of data from an I/O subsystem through the network, possibly subject to some form of real-time constraints. Therefore, efficient storage system designs are fundamental to the viability and success of these applications, and thus we expect our tool to be of use to a broad range of current and future systems and applications.

The main goals in the design and development of ViPEr-HiSS are: (1) identification of proper mechanisms for design specification and subsequent performance evaluation of *hierarchical multimedia storage* systems and incorporation of them into the tool; as well as (2) illustration of how the resulting design tool can aid in the development of novel and useful resource management techniques for storage hierarchies as well as in accomplishing orders of magnitude improvements in performance of future storage technologies.

The remainder of this paper is organized as follows. In Section 2 we discuss challenges and requirements related to multimedia storage systems. In Section 3 we describe the design of ViPEr-HiSS. Section 4 briefly surveys related work. Lastly, in Section 5 we give a few concluding remarks.

2 Storage Systems and Application Challenges

2.1 Application Requirements and Challenges

The characteristics of multimedia applications which lead to difficulties in end-to-end systems design are that they have very large bandwidth and storage needs, with vastly different performance and reliability requirements, often coupled with real-time constraints. Thus, one challenging task in designing multimedia end-to-end systems is satisfying the real-time requirement of continuously transmitting a multimedia object, from the storage subsystem (possibly through a network) to the user.

Providing *on-demand* (or "near-demand") service to multiple clients simultaneously (thus realizing economies of scale) is another challenge. Users expect to access objects within a small and "reasonable" latency upon request, where latency can be attributed to deficiency of resources at any level of the storage hierarchy or the network, at the time the request is made.

Many designs have been proposed that succeed for "small" versions of multimedia applications but simply do not scale up. Thus, another challenge is to properly architect scalable end-to-end systems that will permit the deployment of "large" versions of these applications. Real-time or continuity constraints, large bandwidth, large storage, and concurrent access by multiple users — these are only the tip of the iceberg. The highly interactive nature of a variety of multimedia applications, resulting in fairly unpredictable workloads, presents a whole other challenge. These include various scientific visualization problems, digital effects applications, distributed multimedia warfare simulations, medical imaging applications, 3-D virtual worlds, and many more. Many state-of-the-art systems for such highly interactive applications can only handle memory-based applications. Current needs call for data sets that will not even fit on disk-based systems (e.g., as in [?]).

2.2 Storage System Requirements and Challenges

There are many challenges that multimedia application impose on the storage system design, which result in a need to explore better algorithms for data layout, scheduling, workload characterization, storage device characterization, and construction of data retrieval policy over multiple user requests in heterogeneous (in terms of storage devices, physical media, and applications) environments.

The economics of storage devices are such that there is always a tradeoff between access speed and device and physical media costs. In spite of the dramatic improvements in the cost of secondary storage, high bandwidth magnetic disks alone are inadequate for meeting the storage and delivery needs of many multimedia applications. Thus, *multi-level* storage hierarchies are needed if we wish to take advantage of the low cost of tertiary media but still provide reasonable performance characteristics, in the context of data intensive applications with some form of real-time constraints. (By "tertiary" storage we mean various types of storage devices, such as CD-ROM's, tapes, DVD's, even cheap and slow magnetic disks, for which secondary storage can act as a "cache".) Hence, there is also a need for cost/performance evaluation methodologies that will allow us to properly configure storage subsystems. (Note that, it may not be possible to increase the configuration of a storage subsystem by an arbitrarily small size increment, e.g., it is not always possible to add one or two more tapes to a robotic tape storage library; hence, determining cost-effective configurations is not, in general, a simple task.)

Furthermore, given current advances in network-attached storage devices, there are challenges in designing hierarchical multimedia information systems with storage (and computational resources) that are dispersed across local and wide area networks. Thus, in designs of storage systems we must consider networking resources as well. More specifically, in this paper, we consider System Area Networks (SANs) type environments and designs of storage hierarchies where the various storage devices are connected through a high speed local area network, such as a Gigabit Ethernet. Although issues related to accessing storage devices through Wide Area Networks are of importance as well, they are outside the scope of this paper and are the topic of future research.

It is interesting to note that storage devices are particularly interesting resources, especially in a parallel or distributed environment. Although many basic principles of parallelism do apply, some characteristics are unique to storage subsystems — whereas in the case of resources such as CPUs, one could consider a pool of interchangeable resources whose use can be determined at runtime (at least when dynamic scheduling is used), all storage resources are not equivalent since a storage device's utility is determined by the data stored on it. Moreover, the placement of the data is typically determined a priori. This "partitioning" of resources (based on data placement) contributes to some of the difficulties in designing cost-effective large-scale I/O systems in general. Solutions to this problem, through data layout and scheduling techniques, are often application dependent. The distribution of data among the storage devices of the system can significantly affect the overall performance of that system — inappropriate data placement can lead to load imbalance problems due to skewness in the data access patterns.

It is also important to note that most storage devices can be viewed as a "multidimensional" resources, e.g., for disks the dimensions correspond to storage capacity and bandwidth capacity, where depending on the application one or the other resource can be the bottleneck. Hence, storage related resource allocation problems often correspond to hard theoretical problems [14].

In this section, we have briefly described challenges in storage system designs arising both from application needs and current architectures of storage devices. Much work is still needed on resource management algorithms that can address the above raised issues. However, even with the existence of such algorithms, cost-effective storage system design is no simple task. Thus, in the next section we describe the design of ViPEr-HiSS, a tool that can aid in the design, configuration, and subsequent performance evaluation of hierarchical storage systems, with a focus on multimedia applications.

3 Storage Design Tool

Storage sub-systems are a fundamental part of computer systems and for many data intensive applications they are the main performance bottleneck. Economies of scale, thus far, dictate the use of storage hierarchies, with orders of magnitude differences in storage device (a) capacities, (b) access latencies, (c) transfer rates, and (d) costs. Furthermore, today's storage devices, even those in the same "category", can have vastly different architectures, and thus evaluation of cost/performance tradeoffs in their design and configuration can be quite complex. An excellent example of this are robotic tape storage libraries, as can be seen in Table 1 which illustrates architectural, performance, as well as cost characteristics for some typical robotic tape storage libraries (these are *list* prices which were compiled from quotes, web-sites, and catalogues during the Spring of 1997).

Although there is less diversity in architectural characteristics of magnetic disks, they still differ in storage capacity, transfer characteristics, and cost. Cost is a function of storage and performance characteristics as well as I/O channel protocols used (e.g., SCSI devices are usually more expensive than IDE devices). Moreover: (1) an improvement in one disk characteristic does

Characteristic	ACL 4/52	ACL 6/176	Ampex	Exabyte AME
Max. number of Drives	4	6	4	2
Number of robots	1	1	1	1
Robot Latency (s)	20	20	6	10
Total Latency (s)	210	210	34	195
Max. tertiary bandwidth (MB/s)	20	30	60	6
Tertiary capacity (GB)	1,820	6,020	6,400	400
Cost of Library (w/ 1 drive)	24,000	62,000	280,000	19,375
Cost of Library (w/ max drives)	57,000	117,000	610,000	26,000

Table 1: Robotic Tape Storage Libraries.

not usually correspond to a similar improvement in other characteristics, e.g., it is easier and cheaper to increase storage capacity than to increase transfer rates or reduce seeks, and (2) performances losses due to overheads (such as seeks) are largely a function of the workload characteristics, data layout schemes, vendor implemented optimizations, and so on. (Similar statements are also true for tertiary storage devices.) Thus cost/performance tradeoffs in design and configuration of disk subsystems are not always straight forward as well. (This problem can become worse if we consider disk arrays, as shown in an example of purchasing a RAID array given below.)

3.1 Motivation and Background

Over the years, relatively little attention in system designs has been given to storage system resources, as compared to CPU resources. This is especially true for tertiary storage devices, whose "proper" place in a system's architecture is not well understood [12]. Clearly, the problem is exacerbated further when we consider an entire storage hierarchy design and subsequently try to evaluate its performance.

However, the need for such systems is more pressing every day, due to the enormous storage requirements of modern applications, such as continuous media services, digital libraries, and scientific computing systems. Hence, research on design and evaluation of configurations and resource management techniques at various levels of the storage hierarchy is gaining momentum [23, 18, 22, 29, 1, 17]. All these works illustrate two important points: (1) it is difficult to evaluate and predict performance of storage devices and (2) the required performance characteristics are often achieved at the cost of complex solutions (be they data layout, scheduling, or other algorithms), given the current storage device characteristics and modern application needs. The more complex the architecture and the resource management algorithms (and their interaction), the more difficult it is to predict the overall system performance. This problem is exacerbated by the fact that application designers, which have the need for high performance storage hierarchies, do not usually have the necessary expertise to either employ such complex solutions or to choose among them.

There is a common misconception that storage systems are "cheap". What is getting relatively cheap is the storage space; there is still a great deal of cost in achieving reasonable performance characteristics (such as access latencies) and development (and maintenance) of "custom" systems for high performance applications [5]. Currently existing solutions to large-scale storage problems are expensive, both in terms of (a) software costs (HPSS software costs about $\$100,000$); and (b) people costs (one study [9] reported that "on the average, one full time person was needed to manage every 10G bytes of storage"). The problem is getting worse since the costs associated with storage systems are growing (relative to the cost of the raw storage).

The "build it and then measure it" approach to system design is inappropriate, because after the system is implemented and its performance measured, significant *design* (as opposed to the implementation) changes are largely impossible to make. Therefore, given the vast differences in (a) architectures of storage devices, (b) possible resource management algorithms, and (c) in general, degrees of freedom in the overall system architecture, what is really needed is the capability to (1) predict performance of *designs* and (2) experiment with "what if" type questions. This is precisely the purpose of ViPEr-HiSS.

To give a concrete example of what might seem to be a relatively simple problem of choosing a storage system to buy[1], in [3] the authors describe their experience with in-house evaluation of RAID systems (for the purpose of making purchasing decisions). This system was intended to be used for high energy nuclear physics applications at the Thomas Jefferson National Accelerator Facility. As described in [3], this process involved (a) installing (at the Jefferson Lab) RAID systems from various vendors, (b) running the nuclear physics applications on these RAID systems, and (c) determining whether they achieve the desired performance requirements. This process took approximately one year to complete and required at least one person full time and part of the time of several others.

Imagine how much more difficult, time consuming, and costly such a process would be if a tertiary storage system was involved, or yet a whole storage hierarchy, where the differences between available devices are huge (compared to magnetic disks), and where it is not even clear where in the storage hierarchy, if at all, some devices belong [12]. The need for evaluation of application- and workload-dependent tradeoffs between storage configurations is not uncommon [16, 5], as in video-on-demand systems [2]. Thus design of tools for proper storage system configurations is an

[1]This is not even a design problem, in some sense.

important problem in and of itself. The design and performance evaluation of such systems is becoming more complex, as storage resources are distributed (at least) across LANs, through the use of network-attached storage devices [13]. This type of design will soon become the norm rather than the exception. Thus, ViPEr-HiSS will be useful for design and evaluation of storage systems at a variety of scales.

3.2 Problem Description and Solution

Storage subsystem *design*, especially for large-scale applications with stringent performance requirements, is complex largely due to: (a) a variety of different devices that exist, (b) complexity of their behavior, and (c) degrees of freedom in possible designs. Simply put, storage systems have too many "facets" (in the form of device characteristics, vendor-based optimizations, a mixture of workloads, and a variety of data placement and scheduling algorithms) for a "back of the envelope calculations" approach to be a viable and cost-effective solution to the storage hierarchy design and management problem. Given the (potential) orders of magnitude differences in performance and cost of the resulting systems, it is worth while and often necessary to invest a great deal of effort in their design and configuration. Thus, there is a variety of reasons why a tool for software/hardware co-design of hierarchical storage systems is not only useful but essential. These include:

- system sizing and configuration of the storage system
- customizing the storage system for specific applications and media (e.g., video, audio)
- performance evaluation and prediction of the resulting storage system

The design of storage hierarchies involves a multitude of issues and tradeoffs. These include configuration related issues, such as how many levels should there be in the storage hierarchy, what devices should belong to which level of the hierarchy, how many of each device should there be, how should the devices be configured into groups, how should they be distributed over I/O channels as well as communication networks, which data should (by default) reside on which device, and many more. It also includes application-centric (or media-centric) issues, that influence the overall system performance. These include data layout on the storage devices, migration of data through the storage hierarchy (e.g., prefetching, caching), and scheduling of data retrieval. All the while the design must take into consideration I/O channels and networking resources, the storage resources, and so on.

It is difficult to manage so many degrees of freedom and the resulting complexity, much less make predictions about subsequent system performance. There are two extreme approaches one could take to solving this problem — pure modeling (be it analytical- or simulation-based) or measurement of a system's implementation. Neither extreme is applicable at all stages of the system's design and implementation. What is needed are mechanisms for making a smooth transition from one to the other, as the system design and development progresses. The aim of ViPEr-HiSS is to cover a spectrum of performance evaluation needs which we accomplish partly through the use of combinations of simulated and physical system components, e.g., by starting with mostly simulated devices and making a smooth transition to a complete system implementation.

Figure 1 illustrates the major components of ViPEr-HiSS as well as the interaction between these components. Here, resource management can be viewed as the combination of policy and

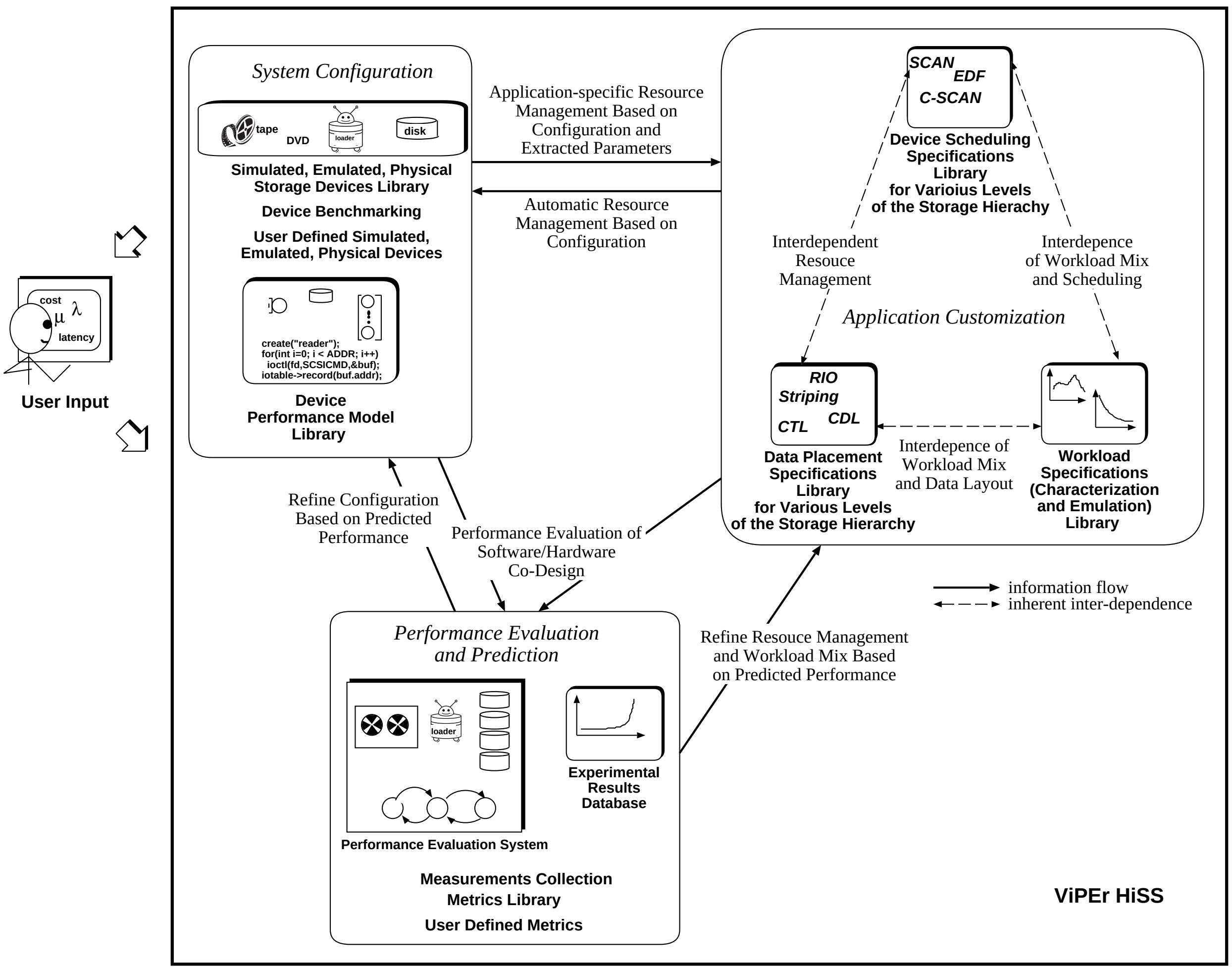

Figure 1: Major Components of ViPEr-HiSS.

mechanism. ViPEr-HiSS is designed to provide a library of common policies, as well as mechanisms to allow definition of new policies. Having a tool infrastructure for low level details allows rapid experimentation with new high level policies as well as their quantitative evaluation. Although different mechanisms may affect the behavior of storage systems, ViPEr-HiSS focuses on policies for the following reasons. First, the slow speed of storage devices relative to processors provides a lot of extra processing cycles, so the speed of the code itself is not as important (conversely, the extra processing power gives the possibility to use computationally expensive policies). Second, using the same mechanism provides a baseline for the comparison of two different policies, otherwise a poorly implemented but good policy may be mistakenly overlooked for a well implemented bad one.

Below we describe each of the components in Figure 1. Note that, in order to make such a tool useful and fulfill the purposes described at the beginning of this section, it must have the following set of essential functionalities[2]:

- common interface for combining physical, analytically modeled, and simulated (or emulated) storage devices;
- data layout schemes including libraries of common layouts and user defined layout;
- scheduling algorithms that include libraries of common scheduling algorithms and user defined scheduling algorithms;
- simulated/emulated storage devices including libraries of common storage devices, user defined devices, and user defined analytical and simulation models of storage devices;
- characterization and evaluation, including benchmarking of storage devices, facilities for workload characterization (and collection of application-related workload data), and facilities for collection of measurements for user defined (as well as predefined) performance metrics.

Work has already been done on some aspects of these requirements, for instance on benchmarking of disks [33], and to a lesser extent tape systems [22, 19]. All of these are needed for a useful design tool. Thus, we incorporate existing techniques into ViPEr-HiSS, as appropriate, and develop our own where the need exists.

[2]Note also that, the goals of ViPEr-HiSS are different from those of designing a multimedia file system mainly, in that it is intended as a *tool* for design, configuration, and performance evaluation of storage hierarchies for specific (collections) of application requirements, rather than a relatively "generic" but modern file system which provides several classes of services to different types of applications (as opposed to the same class of service to all applications, as in a more "traditional" file systems).

We envision several types of uses for ViPEr-HiSS: (1) configuration decisions (both by users and vendors), including decisions on what type and how much of each storage device to include in a storage systems (as in the RAID example above), (2) software/hardware co-design and development of a hierarchical storage system, and (3) research on resource management techniques, particularly in the context of multimedia applications.

3.3 Configuration and System Description

As already stated, proper system configuration can play a huge role in the performance and more importantly cost-effectiveness of a system — poorly sized systems result wasted resources and money. The user first needs to specify the (hardware) configuration or high level architecture of the system. For instance, how many disks the system has, the disk parameters, what types of tertiary devices, and so on. In ViPEr-HiSS, the user can either choose from a library of exististing devices, or define their own, describing the newly defined devices to conform to the tool interface. In either case, device parameters must be specified by the user. We elaborate on the configuration-related functionality next.

3.3.1 Combination of physical and simulated/emulated devices

The need for combining physical and simulated/emulated devices is motivated by the following. First, it is preferable to make as many design and configuration decisions as possible before obtaining all the physical devices that might be needed for the system's development. Second, even when the physical devices are available, it is preferable to evaluate their (potential) contribution to the design before expanding efforts on implementing code for accessing these devices (such as device drivers or raw device interfaces). Thus, a desirable approach is to begin the design and development process with simulated devices[3] and gradually include physical ones, as the system design progresses. As physical devices are added to the system's design, the code can be written.

There is, of course, a need to parameterize the simulated and emulated devices. If a corresponding physical device is not available, then the user can estimate the parameters, by using information obtained from vendors. If, however, the physical device is available, the user can use the benchmarking facilities (described in 3.3.3) to obtain the necessary parameters (since vendor

[3] Note that, performance predictions of initial designs do not have to be very accurate, in a sense that they are needed to make choices between fairly different design possibilities. So, even relatively coarse simulation models can give good initial results and then be refined.

specified information is generally nominal based on optimistic assumptions of use).

In addition to using ViPEr-HiSS to obtain performance predictions, the user might also want to "see" the result of their data retrievals. This is especially useful in multimedia applications such as video-on-demand systems, where lossiness in data delivery is acceptable and the corresponding perceived loss in quality-of-service is often difficult to quantify. In this case, it is desirable for the user to actually see the video played out, even if some of the physical devices are not present. For this purpose, we use "higher" level storage devices to emulate retrieval of data from "lower" level storage devices. For instance, to emulate the retrieval of video from a robotic tape storage library (when we do not actually have one) we can store the video data on a disk; then the physical retrieval will occur from secondary storage, but with performance characteristics (e.g., latency) of tertiary storage. Matching performance is achieved by emulating the characteristics through the use of an appropriate simulation model of the corresponding tertiary device (similarly, an analytical model can be used as well).

3.3.2 Plug-in of user defined simulation/emulation components

This functionality is needed since a tool can not possibly contain all components that might be needed by a particular application. Furthermore, user-defined components can be especially useful for development of systems with some proprietary subsystem components, as part of the overall design. In addition, one can define new components in order to "test" where in a storage hierarchy does a certain type of a storage device belong or what device characteristics would make such as device a "useful" part of the storage hierarchy (e.g., DVD-ROMs). Lastly, a designer can use this functionality in defining performance metrics of interests that are not pre-defined by the tool, i.e., as part of that component's definition (see details below).

As in the case of existing library devices, the user needs assistance in parameterizing user-defined devices — as before, these can either be estimated or obtained through the benchmarking facility, which we describe next.

3.3.3 Benchmarking of storage devices

As already stated, when a physical device is available, there is a need for extracting its parameters (such as seek characteristics, rotational latency, robot latency, and so on) for use in (a) the performance study of the storage system design and/or (b) data layout and scheduling techniques to be developed for a specific application, i.e., there is a need for "benchmarking" physical devices. For instance, there are several proposed data layout technique for video servers [4, 11] that take into consideration the fact that modern disks have multiple physical zones [26]. In order to employ such techniques, one would need to determine details related to the geometry of a disk (e.g., where the physical zones start and end). Benchmarking of the physical devices can also be useful in extrapolating to larger scale designs. Lastly, benchmarking of a physical device can also reveal vendor-implemented optimization, e.g., such as read-ahead of the next track (employed by disk vendors). Such optimizations can be in conflict with user-intended resource management schemes (e.g., read-ahead is often undesirable in scheduling techniques devised for video-on-demand servers [11]), and hence, it would be useful to discover them, and when it is desirable (and possible) turn them off.

Clearly, benchmarking storage devices is no simple task [33, 19], and not one that an application designer might wish to undertake. Thus, we encorporate storage device benchmarking methodologies and provide an automatic benchmarking facility in ViPEr-HiSS.

3.4 Application Customization

Once the hardware configuration is specified, application-related information, such as resource management techniques, must be specified next. Such information, for instance, includes data layout and scheduling techniques. After this step is complete, performance predictions can be made. Depending on the results of the performance evaluation step, the designer may need to repeat some part of the cycle (i.e., make modification either to the configuration, or to the resource management techniques, or both).

3.4.1 Automation of user-defined data layouts

This feature is useful when the designer needs to specify a particular application- or media-dependent technique for layout of data on a physical device. For instance, this is needed by many continuous media servers (e.g., as in [28, 10, 11]) which use a specified layout of data to make performance guarantees. Accomplishing a specific data layout is by no means a simple task, especially since this often results in a need to deal directly with raw storage devices. For the "reasonably well defined" and common data layout techniques, in ViPEr-HiSS, the user only needs to specify a configuration file with appropriate parameters (e.g., GROUPSIZE in the example of Figure 2).

An example configuration file, that corresponds to the Streaming RAID Video-on-Demand server[4] design proposed in [28], is illustrated in Figure 2. This configuration file corresponds to a simple

```
1: simDisk1
2: simDisk2
3: physDisk3
4: ld1    3
5:        simDisk1; simDisk2; physDisk3
6:        (ADDR%GROUPSIZE)
7:        (FLOOR(ADDR/GROUPSIZE))
8:        53067801
9:        1
10:       3
11:!end
```

Figure 2: Example configuration file.

example which requires no code on the user's part[5]. It describes a storage configuration and a data layout for striping an object (e.g., a video) across 3 disks. Lines 1, 2, and 3 are the declarations for two simulated disks and one physical disk that make up the striping group. Line 4 begins the configuration file for a logical disk, namely "ld1" (a logical disk is used to group devices, in order, for instance to specify disk array type configurations) — the "3" on line 4 specifies that "ld1" is made up of 3 component devices. Line 5 lists each of the components (declared earlier). Lines 6 and 7 are the layout configuration functions, each of which maps a logical address (ADDR) to a value — for line 6 that value corresponds to a disk (simulated or physical) and for line 7 that value corresponds to a physical address on a disk. Thus, line 6 gives a function which determines which component disk to store/retrieve data to/from, and line 7 gives a function for determining at which physical address on that component disk the data is stored (and retrieved from). Using these functions, "ld1" can take any logical address and determine on which disk and where on that disk to store data (and subsequently retrieve it). Note that these functions are based on GROUPSIZE (specified on line 10) and so would not need to be rewritten if the number of disks in the stripe group changed. Line 8 gives the size of the logical disk (which is three times the size of each of the homogeneous component disks, in this case). Line 9 indicates that the retrieval from the three disks is to be performed in parallel (e.g., as in a RAID-3 type configuration). Line 10 gives the groupsize, or stripe width, i.e., how many disks participate in a stripe. Line 11 denotes the end of the configuration file.

Finally note that, even if a device is not a physical one (and hence there is no need to actually place the data on the device), it is still useful to specify a data layout technique, since, for many multimedia applications, it has a significant effect on the overall system performance. Again, a

[4] Note that some of the details of cycle-based scheduling proposed in [28] are left out of this example, for simplicity and clarity of illustration.

[5] For ease and clarity of illustration we left out some details of the configuration file specification language.

user can use either an existing configuration file that works with an existing or a user-specified simulated device, or provide his/her own configuration file.

3.4.2 Plug-in user-defined scheduling algorithms

The motivation for user defined scheduling algorithms is the same as that for user defined data layout techniques, to permit customizing the system for a specific application and/or media. ViPEr-HiSS provides this functionality as well. Schedulers, as well as the interaction between them, can be specified at various levels of the storage hierarchy.

3.4.3 Workload characterization

Workload specific information, which is a function of the particular application intended to be run on the system being designed, is needed in order to accurately evaluate and predict the system's performance. Such information is user specified, and includes details such as request arrival rates and corresponding distributions, distributions of skewness in data access, interaction between different applications/workloads, and many more. Thus, in a sense, the application can be emulated as well. For instance, such application emulation techniques have been developed for scientific workloads [30]. We are also developing methodologies for emulation of multimedia applications.

3.5 Performance Evaluation and Prediction

Once the system is specified (i.e., hardware characteristics, resource management techniques, and workload characteristics), the next step is to run/emulate/simulate the system components and collect performance information, in order to determine whether performance and quality-of-service requirements of the applications intended to be run on the designed system can be met. Of course, the user must specify what performance metrics are of interest so that the tool can measure them.

3.5.1 Measurements collection and metric specification

In order to understand the performance of the configuration and design being evaluated with ViPEr-HiSS, it is important to extract performance metrics from the devices in addition to end-to-end statistics. To this end, we are developing a metric description system that allows users to define and compose metrics as part of their system specification. While each device contains simple metrics (i.e., latency, data transfer rates), it is often important to be able to create custom metrics that capture the salient features of a particular target configuration.

For example, consider the case where the system being evaluated is a video server that includes replication of the same video on multiple devices, as in [24]. To evaluate this type of system, a user would like to be able to create metrics that capture the load on each server that has a copy of the video. In addition, they would likely want to know how many requests for the clip were satisfied by each replica. This type of information can be constructed from the basic device statistics, but requires custom metric definitions that track specific video clips rather than individual device operations. ViPEr-HiSS will provide these types of metrics via an interface that allows users to:

1. access statistics about the default devices we supply. Each of our devices will supply a set of standard metrics, and optionally device specific metrics (e.g., time spent waiting for a tape drive to become available in a robotic mass storage library).

2. create new statistics as part of the definition of custom devices. The idea will be to allow users to create any statistics they find useful. To make these metrics useful in composition with other devices, we are defining an interface to export these device specific options.

3. define a simple language to compose the device statistics into more complex metrics. This will likely include simple arithmetic operations and statistics operations such as mean and variance.

4. define a meta data format that allows statistics to be "attached" to the data objects as they pass through each device. By doing this we, can track performance data based on specific requests as they flow through the system.

4 Related Work

An overview of issues and tradeoffs in the design of multimedia storage systems can be found in [15]. Facilitating management of these issues and design decisions based on these tradeoffs is one of the primary motivations for designing and developing the ViPEr-HiSS tool. In the remainder of this section we briefly survey related work on performance evaluation tools and contrast it with our own. Note that, ViPEr-HiSS is an "application-oriented" tool rather than a "methodology-oriented" tool. Thus in the interests of brevity, we will not discuss methodology-oriented tools (such as queueing networks, petri net, Markov chains, and simulation tools) below — a fair amount of literature exists on these topics, which can be found in the Proceedings of the International Conference on Modeling Techniques and Tools for Computer Performance Evaluation as well as at [?]. Similarly, we will not survey the literature on performance evaluation methodology and its application; these can be found in proceedings of conferences such as SIGMETRICS, Performance, and so on.

POEMS (Performance Oriented End-to-end Modeling System) [8] focuses on prediction of the end-to-end performance of *parallel/distributed* implementations of large-scale adaptive applications. As ViPEr-HiSS, POEMS provides a library of component models, from workloads to memory hierarchies, but with a focus on *parallel* application performance prediction. In contrast ViPEr-HiSS focuses on storage systems for multimedia application, with an emphasis on *design* evaluation.

The work of [32] which is implemented in the SPEED tool, has goals similar to those of ViPEr-HiSS (most notably, evaluation of designs) but with a focus on a different application, specifically, that of performance evaluation of *software architectures*. Software performance engineering has also been investigated in the context of client/server systems [25].

The Pablo analysis tool [27] and Paradyn tools [?] have been used for extracting workload characteristics of *parallel scientific applications*. Uysal et al [30], have characterized and modeled I/O workloads of data intensive applications intended for running on large-scale parallel machines for the purpose of their performance prediction. Additionally, in [30, 31] the authors focus on simulation of communication subsystems, mostly for large-scale parallel machines, for application-level performance evaluation; this simulator is configurable over many architectures, using public hardware specifications. In contrast, ViPEr-HiSS focuses on performance prediction of *designs* of multimedia storage systems.

5 Conclusions

This paper presented the issues that arise in design and performance evaluation of storage hierarchies, and specifically with application to multimedia systems. As a solution to the problem of managing complexity of storage hierarchy designs, we are developing ViPEr-HiSS, a performance tool for design, configuration, and evaluation of hierarchical multimedia storage systems. The ViPEr-HiSS framework encompasses many facets of storage system designs, from application workloads through system policies, down to the storage devices themselves. The benefits arising from the design and development of ViPEr-HiSS include: (a) methodologies for system *design* specification;

(b) methodologies for system *design* performance evaluation and prediction; and (c) facilitation of development and evaluation of novel resource management techniques for multimedia applications in the context of *hierarchical* storage systems.

References

[1] David A. Patterson Ann L. Chervenak and Randy H. Katz. Choosing the Best Storage System for Video Service. *In Proc. of ACM SIGMETRICS Conf.*, pages 109–119, 1995.

[2] S. Berson, L. Golubchik, and R. R. Muntz. Fault Tolerant Design of Multimedia Servers. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 364–375, San Jose, CA, May 1995.

[3] I. Bird, R. Chambers, M. Davis, A. Kowalski, B. Lukens, S. Philpott, and R. Whitney. Evaluating raid in the real world. In *Joint Sixth Goddard Conference on Mass Storage Systems and Technologies and Fifteenth IEEE Symposium on Mass Storage Systems*, pages 345–354, College Park, March 1998.

[4] Y. Birk. Track-pairing: A novel data layout for vod servers with multi-zone-recording disks. In *IEEE International Conference on Multimedia Computing and Systems*, pages 248–255, May 1995.

[5] E. Borowsky, R. Golding, P. Jacobson, A. Merchant, L. Schreier, M. Spasojevic, and J. Wilkes. Capacity planning with phased workloads. In *First International Workshop on Software and Performance*, pages 199–207, Santa Fe, Oct 1998.

[6] J. Cowie, D. Nicol, and A. Ogielski. Modeling the global internet. *Computing in Science and Engineering*, pages 42–50, Jan.-Feb. 1999.

[7] H. Eom and J. K. Hollingsworth. Lbf: A performance metric for program reorganization. In *ICDCS*, pages 222–229, Amsterdam, Netherlands, May 1998.

[8] E. Deelman *et al.* Poems: End-to-end performance design of large parallel adaptive computational systems. In *WOSP: International Workshop on Software and Performance. Santa Fe, NM*, pages 18–30, October 1998.

[9] J. P. Gelb. System-managed storage. *IBM Systems Journal*, 28(1):77–103, 1989.

[10] S. Ghandeharizadeh, S. H. Kim, and C. Shahabi. On Configuring a Single Disk Continuous Media Server. *ACM SIGMETRICS Conference*, May 1995.

[11] S. Ghandeharizadeh and R. R. Muntz. Design and implementation of scalable continuous media servers. *Special issue of Parallel Computing Journal on Parallel Data Servers and Applications*, January 1998.

[12] J. J. Gniewek. Towards improved tape storage and retrieval response time. In *Joint Sixth Goddard Conference on Mass Storage Systems and Techn ologies and Fifteenth IEEE Symposium on Mass Storage Systems*, pages 81–94, College Park, March 1998.

[13] H. Gobioff, G. Gibson, and D. Tygar. File server scaling with network-attached secure disks. In *ACM Sigmetrics Conference*, Seattle, Washington, June 1997.

[14] L. Golubchik, S. Khanna, S. Khuller, R. Thurimella, and A. Zhu. Approximation algorithms for data placement on parallel disks. In *To appear in the 11th SIAM-ACM Symposium on Discrete Algorithms (SODA 2000).*

[15] L. Golubchik and R. R. Muntz. *Parallel Database Systems and Multimedia Object Servers.* Springer Verlag.

[16] L. Golubchik, R. R. Muntz, and R. W. Watson. Analysis of Striping Techniques in Robotic Storage Libraries. *Proc. of the 14th IEEE Symposium on Mass Storage Systems*, pages 225–238, September 1995.

[17] L. Golubchik and R. K. Rajendran. A Study on the Use of Tertiary Storage in Multimedia Systems. *In Proceedings of the Joint NASA and IEEE Mass Storage Conference*, March 1998.

[18] B. K. Hillyer and A. Silberschatz. Scheduling non-contiguous tape retrievals. In *Joint Sixth Goddard Conference on Mass Storage Systems and Techn ologies and Fifteenth IEEE Symposium on Mass Storage Systems*, pages 113–124, College Park, March 1998.

[19] Bruce K. Hillyer and Avi Silberschatz. On the modeling and performance characteristics of a serpentine tape drive. In *Proceedings of the 1996 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, Philadelphia, PA*, pages 170–179, May 23-26 1996.

[20] J. K. Hollingsworth. Critical path profiling of message passing and shared-memory programs. *IEEE Transactions on Parallel and Distributed Computing*, 9(10):1029–1040.

[21] J. K. Hollingsworth, B. P. Miller, M. J. R. Goncalves, O. Naim, Z. Xu, and L. Zheng. Mdl: A language and compiler for dynamic program instrumentation. In *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 201–212, San Francisco, Nov 1997.

[22] T. Johnson and E. L. Miller. Benchmarking tape system performance. In *Joint Sixth Goddard Conference on Mass Storage Systems and Techn ologies and Fifteenth IEEE Symposium on Mass Storage Systems*, pages 95–112, College Park, March 1998.

[23] J. Kekashman. Building and managing high performance, scalable, commodity mass storage systems (poster paper). In *Joint Sixth Goddard Conference on Mass Storage Systems and Techn ologies and Fifteenth IEEE Symposium on Mass Storage Systems*, pages 175–180, College Park, March 1998.

[24] P. W. K. Lie, J. C.-S. Lui, and L. Golubchik. Threshold-Based Dynamic Replication in Large-Scale Video-on-Demand Systems. *In Proceedings of the Eighth International Workshop on Research Issues in Data Engineering: Continuous-Media Databases and Applications (RIDE'98)*, February 23-24, 1998.

[25] D. Menasce and Hassan Gormaa. On a language based method for software performance enginerring of client/server systems. In *WOSP: International Workshop on Software and Performance. Santa Fe, NM*, pages 63–69, October 1998.

[26] Chris Ruemmler and John Wilkes. An Introduction to Disk Drive Modeling. *IEEE Computer Magazine*, pages 17–28, March 1994.

[27] Evgenia Smirni and Daniel A. Reed. Workload characterization of input/output intensive parallel applications. In *Proceedings of the Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Springer-Verlag Lecture Notes in Computer Science*, pages 169–180, June 1997.

[28] F. A. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming RAID - A Disk Array Management System For Video Files. *ACM Multimedia Conference*, pages 393–399, 1993.

[29] P. Triantafillou and T. Papadakis. On-demand data elevation in a hierarchical multimedia storage server. In *Twenty-third International Conference on Very Large Data Bases*, Athens, Greece, August 1997.

[30] M. Uysal, Tahsin Kurc, Alan Sussman, and Joel Saltz. A performance prediction framework for data intensive applications on large scale parallel machines. In *Fourth Workshop on Languages, Compilers and Run-time Systems for Scalable Computers (LCR'98)*, 1998.

[31] Mustafa Uysal, Anurag Acharya, Robert Bennett, and Joel Saltz. A customizable simulator for workstation networks. In *Proceedings of the 11th International Parallel Processing Symposium, Geneva, Switzerland*, April 1997.

[32] Lloyd Williams and Connie Smith. Performance evaluation of software architectures. In *WOSP: International Workshop on Software and Performance. Santa Fe, NM*, pages 164–177, October 1998.

[33] B. L. Worthington, G. R. Ganger, Y. N. Patt, and J. Wilkes. On-line extraction of scsi disk drive parameters. In *ACM Sigmetrics Conference*, Ottawa, May 1995.