HRaid: a Flexible Storage-system Simulator *

Toni Cortes Jesús Labarta
Universitat Politècnica de Catalunya - Barcelona
{*toni, jesus*}*@ac.upc.es* - *http://www.ac.upc.es/hpc*

Abstract
Clusters of workstations are becoming a quite popular platform to run high-performance applications. This fact has stressed the need of high-performance storage systems for this kind of environments. In order to design such systems, we need adequate tools, which should be flexible enough to model a cluster of workstations. Currently available simulators do not allow heterogeneity (several kind of disks), hierarchies or resource sharing (among others), which are quite common in clusters. To fill this gap, we have designed and implemented HRaid, which is a very flexible and easy to use storage-system simulator. In this paper, we present this simulator, its main abstractions and some simple examples of how it can be used.

Keywords: Storage-system modeling and simulation, Heterogeneous RAIDs, Cluster computing

1 Introduction

One of the main characteristics of a cluster of workstations is its ability to grow whenever needed. If more storage space is required, a new disk can be easily attached. This ends up making it very difficult to have an homogeneous set of disks, unless all disks are replaced when a new one is bought. All simulation tools that are currently available assume a set of homogeneous components [1, 2, 3]. This kind of tools is adequate to simulate storage system for parallel machines, but it lacks the flexibility to study the performance of an I/O system for a cluster of workstations.

It is also quite common, in a cluster, to have plenty of resource sharing within a storage system. Most tools do not take into account the effect of network or CPU contention. These issues cannot be left aside as the performance may depend on them heavily.

We believe that it is very important to have tools that allow researchers to see the effect of a given policy using more real models than the ones currently available. HRaid is a new tool that gives the possibility to researchers to model heterogeneous storage system such a RAIDs where the disks are not homogeneous. It can also be used to model hierarchies of RAIDs or any other kind of network-distributed system due to its great configuration flexibility. Of course, it also allows to model traditional environments with homogeneous components, or even single disk systems. Furthermore, it is also flexible enough to allow any researcher to add new components or new models for already existing components (such as disks or networks).

In this paper, we will present the main functionalities of this new tool and the components that have already been implemented, which should be enough for most research works.

1.1 Kind of Problems to be Addressed Using HRaid

To get a better idea of how this new tool could be used, we present a few example questions that can be answered using it.

- What is the effect of heterogeneity in a network RAID? How is the performance affected if a new and faster disk is added to the RAID system? Is it worth to buy

*This work has been supported by the Spanish Ministry of Education (CICYT) under the TIC-98-0511 contract.

the fastest disk or with a slower one the same performance will be achieved? Are there better placement algorithm to be used when a RAID is not homogeneous? Which ones are them?

- How important is to place files in the disk which is closer to the application using it?
- When partitioning files (as done in Vesta [4] or PIOUS [5]) among heterogeneous disks, how should the sizes of the partitions be?
- How important is the network when implementing a distributed storage system?

1.2 Related Work

As far as we know, there are no tools as flexible as the one we present in this paper. All the simulators we have found are very specialized ones. They simulate single disks, a set of independent disks, or even RAIDs, but all of them can only model very specific and pre-defined storage systems. Our simulator no only models RAIDs but can be used to simulate any kind of storage system. It can simulate network attached I/O systems, disk sharing among nodes, hierarchical storage systems, etc. Furthermore, it can simulate very detailed disks and large storage systems, all at the same time, which allows much more reliable results.

Ruemmler and Wilkes presented a disk model [6] that is widely used in many disk simulators. This model is the one we have used for our disk module although we are planning to improve it and adapt it to new concepts such as varying data density. The disk simulator implemented by Kotz [2] is a good example of a simulator that follows this model.

RAIDFrame [1] is framework for rapid prototyping and evaluation of redundant disk arrays. This tool models RAIDs as acyclic graphs, which is a quite intuitive way to specify them. The problem with this tool is that it is specially designed to work with RAIDs and that no simple extension can be done to implement heterogeneous RAIDs or more complex storage-systems.

Finally, the tool that is closer to the one we present in this paper is DiskSim [3]. DiskSim is a high-configurable and very detailed disk system simulator. Although many storage systems can be configured, systems such as heterogeneous RAIDs are not possible (at least in the current version). HRaid is more flexible, powerful and easy to use, but DiskSim has a more detailed disk model, which probably makes it more accurate. Anyway, we plan to improve our modules to match their accuracy in a short period of time.

2 Simulation Model

2.1 Model Abstractions

In this section, we will describe the abstractions used to model a storage system for a cluster of workstations in HRaid.

Drive: Disk & Controller

Any storage system must have a place where the data is permanently stored. The **disk** is the abstraction offered by HRaid to perform this task. Although we use the term disk, this abstraction refers to any device that can be used to store permanent data as could be a disk, a tape, a CD-ROM, etc.

Disks need a processing unit, or controller, to manage their behavior. The controller has to position the head on the correct block and to request the transfer. It is also in charge of the communication between the disk and the upper-level requester, and in many cases it manages a small cache to increase the performance. To model this component, we have defined the **controller** abstraction, which represents the hardware card that controls a disk or a tape.

As a disk always needs a controller to be able to work, we have grouped them in a new abstraction named **drive** that connects a controller to a disk. We have decided to offer separated abstractions for disks and controllers to increase the flexibility of HRaid. In this way,

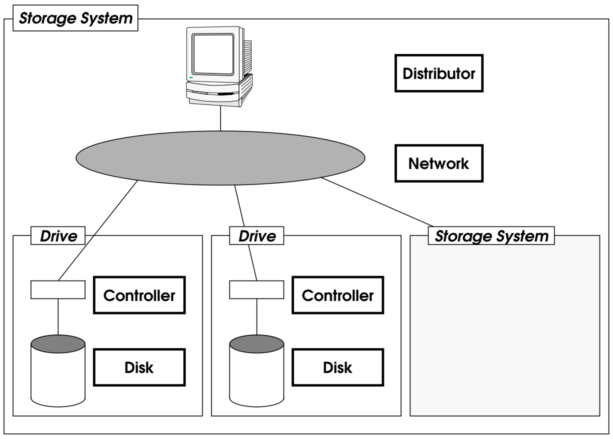

Figure 1: Schema of all the abstractions offered by HRaid.

we can study what would happen if a given disk was managed by several different controllers. A schema of how these abstractions are related can be seen in Figure 1.

Storage System: Distributor & Network

As this tool is specially aimed at studying the behavior of storage systems for clusters of workstations, we want to be able distribute data among disks emulating any system one can think of. We want to be able to simulate RAIDs, systems that place files in the disk that it is closer to the node that most frequently use them, or even file partitioning as done by PIOUS [5] or Vesta [4]. This software/hardware which is in charge of distributing data among disks is what we have called a **distributor**. A parallel/distributed file system would be a software distributor while a RAID card controller would be a hardware distributor. Our simulator does not make any difference between hardware or software as they are only two different ways to implement a distribution algorithm.

For a distributor to be able to use several disks, it needs some kind of **network** to distribute the data onto those disks. This network can either be a SCSI bus, an Ethernet or any other hardware that could be used for communication. This is another of the abstractions we propose.

As a distributor will always need a network to distribute the data among the different disks, we have grouped these two abstraction in what we call a **storage system**. A storage system, besides distributor and network, also needs to have a set of *drives* in where to keep the data (Figure 1).

Storage Devices

While describing the storage system, we have assumed that the data is distributed among drives. This is not necessarily so as we could also want to distribute the data among other storage system making some kind of hierarchy. For this reason, we say that a storage system distributes its data among **storage devices** that can either be a drive or a storage system (Figure 1).

2.2 Simulation and Modules

Now that we have gone over the abstractions offered by HRaid, it is a good time to see how the simulator works. The main function of HRaid is to manage events that go from one instance of a given abstraction to another one. It takes track of the time and delivers events, whenever they are triggered, to the correct abstraction instance. The rest of the code is made of modules that implement instances of any of the following abstractions: disk, controller, network or distributor. Of course, the current version of HRaid has the most common modules already implemented and ready to be used (Section 2.4).

Modules are an implementation of a given model for one of the abstractions. For instance a SCSI-bus module is an instance of the network abstraction. In order to be able to plug modules together, the code has to conform the following simple rules:

- A module can only interact with other modules through a pre-defined interface. This interface allows modules to send/receive data to/from other modules and to request certain operations from other modules. In table 1, we can see this interface between modules.

Abstraction that has it	Abstraction that uses it	Function
Disk	Controller	`request_data`
Controller	Network	`deliver_request`
Network	Distributor	`send_request`
	Distributor	`send_reply`
	Controller	`send_reply`
	Distributor	`grab_network`
	Controller	`grab_network`
	Distributor	`release_network`
	Controller	`release_network`
	Distributor	`send_data`
	Controller	`send_data`
Distributor	Network	`deliver_request`

Table 1: Interface between modules.

- A module can only insert and extract events for itself.
- All modules must have an entry function that it is called by the event manager whenever an event for that module is triggered.

As we can see, modules are quite independent one from the other. This simplifies the programming of new modules as no special care has to be taken to be able to use new modules with older ones. This freedom of connection between modules is what makes HRaid so flexible that it can model nearly any kind of storage system.

2.3 Some Examples

To clarify how these abstractions can be used, we present a set of storage systems and how they are modeled using HRaid.

Heterogeneous Network RAID

In this example, we have a cluster of workstations where nodes are connected through a bus-type network. We also have four disks attached one to each node. Among these disks, we have two fast and two slow ones. In this environment (shown in Figure 2), we want to build a network RAID level 5. HRaid allows us to model this storage system and to study the effect that the load unbalance may have on the performance. It can also help us to see the effect of sharing the bus, etc.

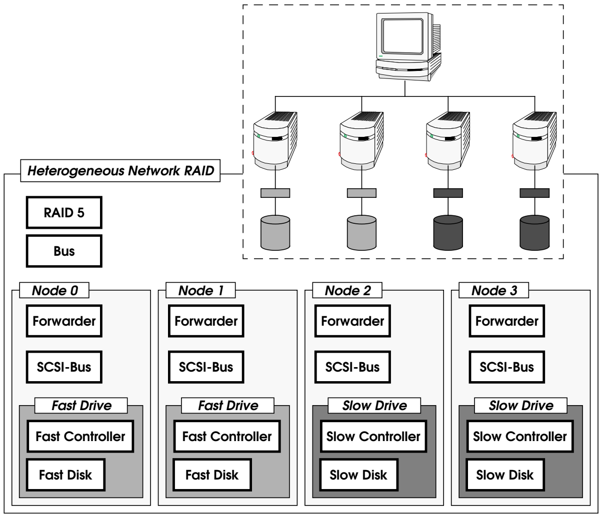

Figure 2: Heterogeneous Network RAID.

To model this storage system, we need two distributor modules: one RAID level 5 and one forwarder, which only forwards the request it receives to its attached disk. We also need two network modules: one SCSI bus and one Ethernet bus. And finally, we need the slow and fast controller and disk modules.

Network-attached RAID

In this second example, we have a network-attached RAID that is part of two independent storage systems. Both hosts share the RAID that it is connected to a shared bus. Figure 3 presents a schema of this environment.

To model this environment we need to define two storage systems (SS1 and SS2) that are connected to a RAID through a bus. Both the bus and the RAID are shared and this is specified by giving them a name (SBus and SNAR), which is used in both storage-system definitions.

RAIDs level 10 and 53

Besides the typical RAID levels (0-5), a combination between basic levels has also been proposed. For instance a RAID level 10 is a RAID level 0 where each segment is RAID level 1.

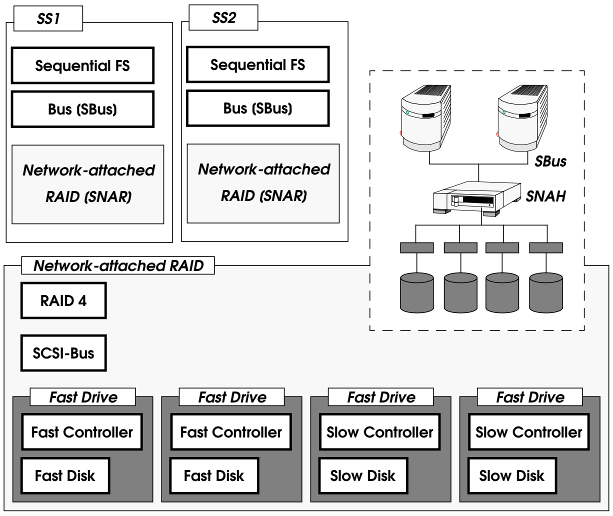

Figure 3: Network-attached RAID.

This kind of combinations are extremely easy to model in HRaid. We only have to configure a RAID level 0, where each device system is a RAID level one. In a similar way, we may define a RAID level 53 which is a combination between levels 3 and 0.

Although this is not necessarily a network example, it shows the flexibility of our tool. All pieces can be mixed together to make more complicated system with very little (or none) effort.

2.4 Implemented Modules

So far, we have implemented the most common modules to be able to start using the simulation to perform some research. In this section, we describe them and present the parameters that can be used to modify their behavior.

Disk Controller and Drive

The disk drive and the controller implemented follow the definition proposed by Ruemmler and Wilkes [6]. This is a quite detailed model of disks and controller which is currently being used in most disk simulators. In Tables 2 and 3 we present the parameters needed to model a controller and a disk as proposed in the afore mentioned paper.

Physical geometry	
	`sector size`
	`sectors per track`
	`tracks per cylinder`
	`cylinders`
Transfer rate	
	`revolutions per minute`
Seek/search overhead	
	`track-switch overhead`
	`track skew`
	`cylinder skew`
	`head-movement overhead`

Table 2: Disk parameters

CPU consumption	
	`new-command overhead`
Cache information	
	`block size`
	`cache size`
	`read ahead (Yes/No)`
Transfer information	
	`read fence`
	`write fence`
	`immediate report (Yes/No)`
	`done message size`

Table 3: Controller parameters

Bus

The only network that has been implemented so far is a bus-based network. We believe that this is the most general kind of network and will be most useful in many cases. The way we have modeled the bus is a simple, but widely used, one based on latency and bandwidth. The latency models the time needed to start a communication while the bandwidth is used to compute the real time needed to send a message through the bus.

This bus only allows one message at a time, which simulates the network contention that can be found in a real net. This is very important as it can make a big difference if all the disks in a RAID are connected through a BUS, which is the most common case [7].

Sequential File System (or forwarder)

This module is basically offered for users that only want to work with a single disk or a hardware RAID. The only thing it does is to receive

CPU consumption	
	`new-command overhead`
Transfer information	

	`block size`
	`immediate report (Yes/No)`
	`request message size`

RAID level 1	
	`read policy`
RAID levels 4 & 5	
	`parity-computation overhead` `small-write policy`

Table 4: RAIDx parameters

requests from the trace file and forwards them to the attached device though the network. We could not use a single disk without this module because then we would have no way to model the bus that connects the host to the controller.

RAID

Finally, we have implemented four modules that emulate the behavior of RAIDs levels 0, 1, 4 and 5 as described by Chen et al. [7]. It is important to notice that all these RAID can distribute the blocks either over drivers or over other storage systems. It is also very important to realize that RAIDs do not need to be homogeneous: we can have a RAID with two fast disks and four slow ones.

The possible parameters used to define the behavior of a RAID are presented in Table 4. Among all of them, I would like to explain three of them that my not be clear just by their name. The first one is the `read policy` needed for RAIDs level 1. This parameter tells the module how to chose between the two copies of a block in a read operation. The two implemented options are *random* and *shortest-queue first* [8]. For RAIDs level 4 and 5 we have also implemented two `small-write policies`: *read write modify* and *regenerate write* [8]. Finally, we have also allow `immediate reporting`, which means that the distributor will send a done message as soon as all data is in its buffers, even if no data has already been physically written.

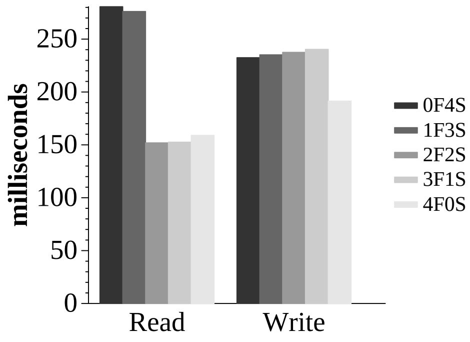

Figure 4: Read and write performance for a RAID1.

2.5 Model Validation

All modules already implemented have been validated by comparing their results to the ones obtained experimentally from real systems. The disk is the only exception as we are trying to obtain real traces from a real disk. Anyway, we compared it to the simulator already implemented by Kotz [2]. When comparing both simulators all requests differed less than 0.2%. As this simulator was already validated, it also validates our implementation.

3 Using HRaid in Simple Examples

To illustrate some of the issues that can be studied with this new tool, we present two simple, but interesting, examples. Figures 4 and 5 present the performance of a RAID1 and a RAID5 configurations while varying the kind of disks used to build them. In both examples, we used two kinds of disks: slow and fast. The fast disk was nearly twice as fast as the slow one. Each bar in the graph represents a RAID configuration varying the number of fast disks (F) and slow ones (S).

In Figure 4, we can see that in a RAID level 1, changing slow disks by fast ones, only makes sense if you change half of them. In that case, read operation are improved quite a lot as the fastest disks are used for reading. On the other hand, write operations are not improved by changing slow disks by fast ones because all

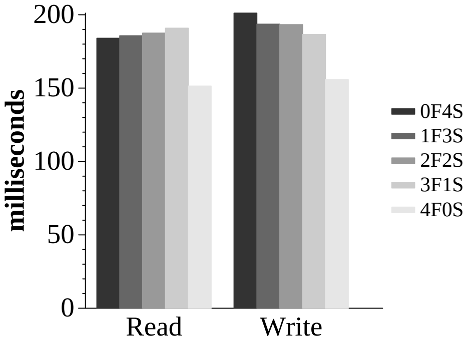

Figure 5: Read and write performance for a RAID5 (regenerate-write for small writes).

disks are needed to perform a write operation. Actually, the performance is decreased. This happens because fast disks tend to use the network more greedily while the slow ones, which are the bottleneck, have to wait for the network. This increases their response time, decreasing the overall write performance.

The results of performing the same experiment in a RAID level 5 is shown in Figure 5. On read operations we can see the effect of network contention as explained for RAIDs level 1. On the other hand, write operations are improved when fast disks are placed because all the small writes that can be done on only fast disks, are done much faster.

4 Conclusions

In this paper, we have presented a new tool aimed at simplifying the task of researching on storage systems for clusters of workstations. We have presented its main abstractions, its functionality and some simple examples of how it can be used.

Acknowledgments

I would like to thank David Kotz for allowing us to use his disk simulator. We learned a lot by examining and using it.

References

[1] W. V. Coutright, G. Gibson, M. Holland, and J. Zelanka. A structured approach to redundant disk array implementation. In *Proceedings of the International Computer Performance and Dependability Symposium*, September 1996.

[2] D. Kotz, S. B. Toh, and S. Radhakrishnan. A detailed simulation model of the HP-97560 disk drive. Technical Report PCS-TR94-220, Department of Computer Science, Dartmouth College, July 1994.

[3] G. R. Ganger, B. L. Worthington, and Y. N. Patt. *The DiskSim Simulation Environment. Version 1.0 Reference Manual.* Department of Electrical Engineering and Computer Science, University of Michigan, February 1998. (Technical report number CSE-TR-385-98).

[4] P. F. Corbett and D. G. Feitelson. The Vesta parallel file system. *ACM Transaction on Computer Systems*, 14(3):225–264, August 1996.

[5] S. A. Moyer and V. S. Sunderam. PIOUS: A scalable parallel I/O system for distributed computing environment. In *Proceedings of the Scalable High Performance Computing Conference*, pages 71–79, 1994.

[6] Chris Ruemmler and John Wilkes. An introducction to disk drive modeling. *IEEE COMPUTER*, pages 17–28, March 1994.

[7] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-performance and reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, 1994.

[8] S. Chen and D. Towsley. A performance evaluation of RAID architectures. *IEEE Transactions on Computers*, 45(10):1116–1130, October 1996.