M.I.T. Laboratory for Computer Science

Request for Comments No. 218
March 19, 1982

Remote Virtual Disks for Vaxen

by Michael Greenwald

Disclaimer

This document does not represent a proposal that is meant to be binding on all owners of VAXen at MIT. It is only a proposal describing what we (Computer Systems Groups) intend to do. Anyone else is free to go along - this will not affect your VAX if you don't want it to.

1. Description: What we have

In about six weeks DEC will deliver 24 VAX 11/750s to be used as personal computers. These will each have 2M main memory, and an RK07 Disk drive having roughly 28M of storage. The RK07 does not have a removable pack. Each VAX will also have a network interface - either a 3COM 10Mb EtherNet driver, or a 10Mb v2lni driver. (Currently, for ours, the v2lni seems more likely - there appears to be more software available for it.)

Initially the VAXen will be running VAX Unix (Berkeley V4.1).

2. The Problem: What we want

The VAXen are meant to be used as personal computers (whatever that is supposed to mean). In order to use them as such we have to overcome several problems.

~ There are more people who would like their own personal VAXs than there are VAXen.

~ VAXen are hot, noisy, and bulky. They really want to be in a physically distant place - we are not going to put them in offices. They will all probably be kept together at some presently unknown location.

~ We have a reasonable total disk capacity, but people would like their personal computer to have many disks of many sizes. Some people can be satisfied with a single 5Mb disk, while others might need several 50M disks. Currently every VAX has a single, non-removable, 28M disk drive.

What do we mean when we say that we want to be able to use the VAXen as personal computers? Well, we would like you to be able to walk over to any terminal that can play Telnet and attach oneself to any free VAX. We would like you to be able to spin-up any number of your virtual disk drives, and put any one of "your" (i.e. disks you can access) virtual disks into the drive - without operator intervention.

If you are using UNIX you can then *mount* these disks into the file system. In any case you can treat them as devices.

Basically, we are trying to maintain the illusion that you have a VAX processor and any number of disk drives all your own. Also, you are able to walk over to your closet, select a disk pack, and place it in one of your drives. No more, and no less.[1]

2.1. Important Note

Please note that this is *not* intended to be a remote file server.

This is *not* intended to be a hack to enable sharing. (Although, just as one can implement sharing on top of existing physical disks, one can presumably implement shared disks/files/objects on top of this.)

[1]Actually, you do get a little more, since now we can share disk packs more easily. So, for example, the CLU library can be a separate disk, and can be shared in READ-ONLY, so that people don't need to copy it onto their own disk.

This is simply a "virtual hardware" extension.

3. Method: What we are going to do

Our proposed plan of action is to implement a Remote Virtual Disk Server, that will allow us to treat the "virtual disks" as if they were physical disks connected to the VAX.

The Remote Virtual Disk Server will be a machine controlling a large quantity of long term memory. A "virtual disk" will be an abstraction supported on the server that will look, to the outside world, exactly like a disk in a drive.

All transactions with the virtual disk will be transacted through a datagram protocol over a network. This includes spinning up the disk, setting the read-only, read-write switches, reading blocks of data, and writing blocks of data.

Additionally, there will be higher level management of the virtual disks. This will include the creation and deletion of disks, the setting of disk passwords, and the renaming of disks.

3.1. The Server

The server will be implemented on a seperate machine. Presumably, that machine will also be a VAX. It will possess some collection of disk drives, the sum of whose memories will be large compared to a single RK07. We do not presently have a good guess as to what the server will look like. The server will accept requests and respond. It can spew forth blocks of data, and will acknowledge the receipt of data - after the data is written to disk successfully. It will return error messages in case of failure.

Planned enhancements to the server include verify-write requests (write and return information about the write. This might consist of a write immediately followed by a read), chained commands, and simultaneous writes to many virtual disks. We would also like to do something to allow the writing of contigous pages to proceed without undue thrashing. If you don't know in advance that you are going to be receiving consecutive pages then odds are, you will only be able to store 1 page to disk per revolution.

The server also provides gross mechanisms for sharing of virtual disks. You can specify, when you spin up a disk, whether anyone else will simultaneously be allowed to access that disk.

3.2. The Protocol

We are going to define a very simple datagram protocol that will access the server. It will be able to do some very primitve things.

~ It will be able to spin up a virtual disk.

~ It will be able to set any physical switches on a drive (read/write/etc.)

~ It can read some number of pages.

~ It can write a page.

~ It can remove a virtual disk from the drive.

This protocol will be a datagram protocol, layered on top of the DOD IP protocol. Eventually we might just use it on top of raw local net packets. The most compelling argument for layering it on top of IP for the time being is that for testing, we can implement a hack server anywhere in the Internet, and have packets get there. During testing, delay isn't all that important. We enclose a checksum in each packet. Future modifications might make more extensive use of this (especially in the verify-write request).

Data is read/written from/to the virtual disks in fixed size blocks, or pages, of 512 bytes (8 bit bytes) each.

Authentication is a problem, but as a first pass we will merely require a password when you spin up a disk. There will be different passwords (or capabilties) for each mode that you access the disk. (i.e. there will be one password for read-only, and another for read-write).

When you spin-up a disk, you are required to identify the virtual-disk that you wish to spin up, an identifier (drive number) that is unique for your host, a mode, and a password for that disk in that mode.

If everything is correct, the server will respond with an ACK, and from then on, you simply use the drive number to refer to your virtual disk. In the case of an error, the server will respond with an error packet identifying the error.

Spinning down the disk is simpler. You simply identify the drive you wish shut down, and the server converts that to the approriate virtual disk. Subsequent references to that virtual disk will result in an error.

To read data from the disk, you specify drive, starting block number, and the number of blocks you wish to read. The read request, just as a physical disk, can time out. If it does not time out then the block is returned. (If more than one block was requested, they are returned in seperate packets. Each packet is of fixed length) Error information is returned in a 32 bit status word. If there is an error, the data may be undefined.

A write request includes the drive, the block number, and the block of data. A WriteAck includes the drive, the block number and a 32 bit status word.

We are tacitly assuming that the server will be on the same local network as the VAXen. This assumption led to the following:

~ We assume that the net is reasonably reliable.

~ We asssume that packets will not get spuriously duplicated.

~ We assume that round trip delay will not be extravagant.

This is the current extent of the protocol. For more detailed information, see *VAX Remote Virtual Disk Protocol.*

3.3. Integration Into The System

The first virtual disk user program will be implemented under VAX Unix as a device driver. We hope to be able to convince UNIX that a virtual disk is a piece of hardware just like an RP06, or an RK07. The present plan is to write a device driver for UNIX for a Virtual Disk that does all network protocol stuff under the covers. We expect to share the current IP code inside the UNIX kernel.

The local disk will be a seperate device. We will store a vanilla-unix on the local disk, so that we can boot from it, and save some initial transfers.[2] We will use the local disk for swapping space.

Again, note that efficiency is not our main goal - we are willing to sacrifice *some* speed for the flexibility that this scheme offers. Bascially, if the virtual drives are almost as fast as the RK07s we will be satisfied.

We have no specific plan, yet, for what the server itself will look like - whether it will be a bunch of RK07s, RM05s, Tridents, or anything else.

3.4. Management of Virtual Disks

The management of these virtual disks is a completely independent task. It can be managed by a TELNET connection to the Server, or by physically going to the Server and messing around. The point is - it can be completely seprate from the protocol we developed here.

For the time being, creation of virtual disks will be channeled through a single administrator. This administrator will also be able to set the passwords on virtual disks.

Deletion, renaming, and listing of disks that you have access to, should be allowable for anyone. As long as you have delete access to a virtual disk, you can rename or list (or, obviously, delete) it.

4. Schedule

4.1. Short Term Plans

1. Design the protocol.

2. Implement a device driver, a Virtual Disk user, on the VAX.

3. Implement a hack server, anywhere.

The short term goals should be achieved before the delivery of the VAXs (Mid April).

[2] You can still use this protocol, and not store UNIX on your local disk, although this makes your VAX incompatible (in the sense that you can tell the difference when you grab this VAX) with those that do store UNIX. You can even run UNIX on a VAX that doesn't have UNIX on its local disk - as long as that disk has some version of this protocol implemented. You can then spin-up a vanilla UNIX as a virtual disk, and run from there. The local storing of UNIX is just a convenience.

4.2. Longer Range Goals

When we have more developement time, we must plan the "permanent" server to take the place of the quick and dirty one we use for testing.

There are enhancements to the protocol that we would like to make.

There is always room for performance improvements.

When we have the design of the server down, we have to pass this information around to allow users to implement "intelligient" disk drivers that know how to queue things approriately for virtual disks.

Eventually we want to automate the virtual disk management facilities, and possibly augment them.

Presumably other problems will crop up as we use them.