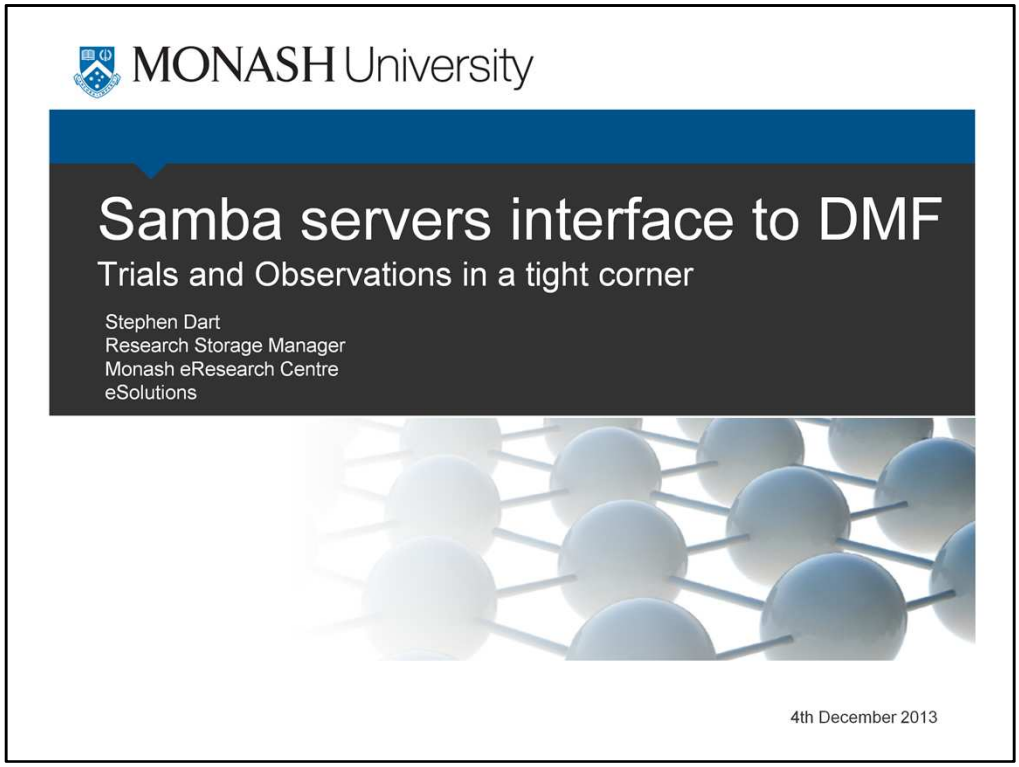
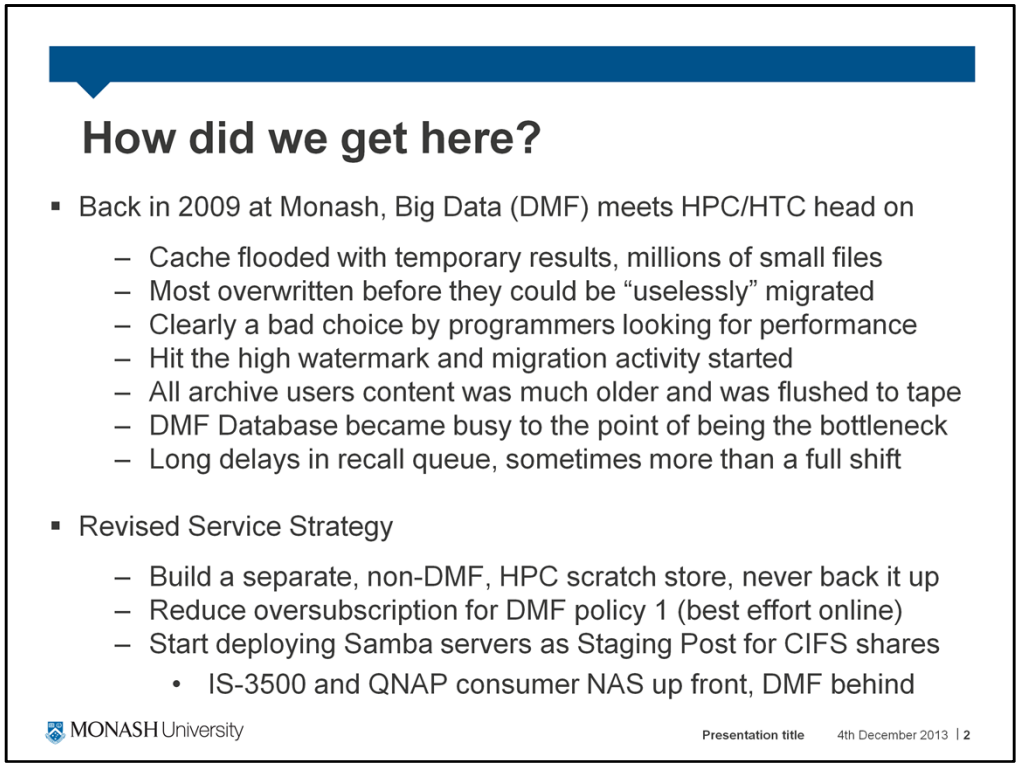


Since I joined the Monash eResearch Centre as the LaRDS Service Manager in 2009 I had a number of serious issues to address.

The archive was growing, but performance was going down. All users complained that LaRDS was slow. In reality, just some of the data was offline and the time to recall it was dramatically variable. Users would not notice that everything was worked as expected. They would focus on those things that stalled. Because the resource of LaRDS was shared among many, recall times were subject to unseen influences. No user could form any kind of consistent expectation from the history of their experiences.

User expectations could not be managed by any kind of elaborate technical explanation. Blah, blah, blah, tape, blah, blah, blah, slow.

As I was digging into the combination of things that were throttling the service, a major incident occurred.



Read the slide text first…….

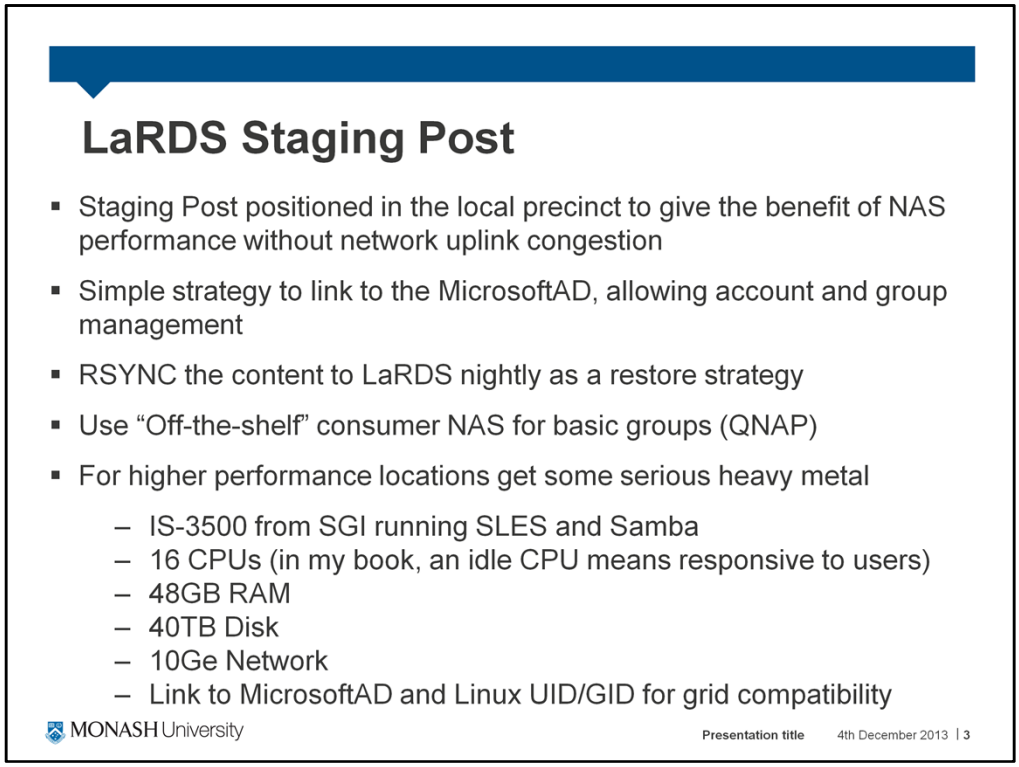
Essentially we found a pretty common way to use a grid of processors to DDOS ourselves into a stupor.

Included in this mix was a number of VMs that were providing CIFS share access using Samba to re-share a NFS mount from LaRDS DMF. These VMs were spread across four host machines, all serving 50 VM instances through a single 1Gb network port. Just about everything went into IOWait and nothing got done.

Not only was performance extremely slow, but the use of NFS re-share had completely removed any capability to forward an indication of which files were online or offline. The IOWait state continued for days!

A multi-pronged approach was used to address the issues and a completely revised service strategy formed.

The solution was to get higher bandwidth and to remove the VM I/O bottleneck. If at all possible, have CIFS shares directly from the DMF host so online/offline status could be displayed.



The LaRDS Staging Post began to replace the VM based Samba servers.

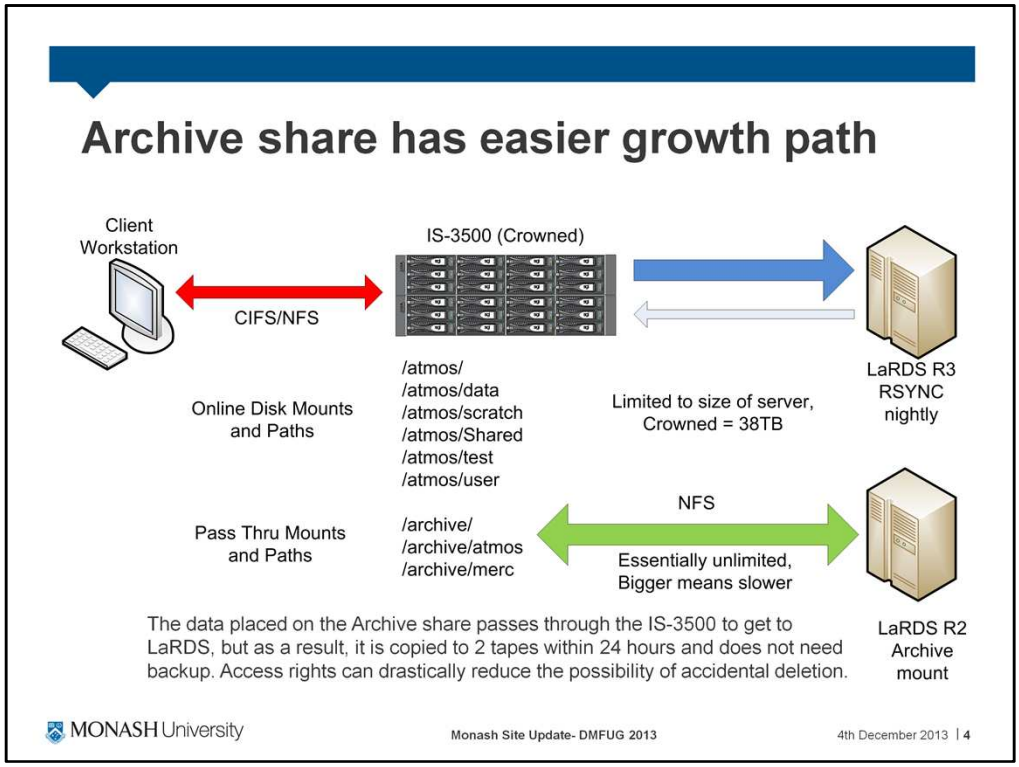
Many new groups and users were able to make use of the CIFS service by authenticating with a corporate managed username and password.
With an attachment to the AD, groups and ACLs had a built-in service management infrastructure, including Service Desk and fault reporting.

Very few new users still required NFS connection to their desktop workstations. As we were/(and are still) running NFSv3, a fixed IP address was required and the user mount had to be root, squashed. We do not have control over the alignment of user workstation managed UIDs.

Also in this mix, the networks team was busy moving all the IP address allocation through a Dynamic DHCP reconfiguration, and getting the same IPv4 address became increasingly difficult. It was hard enough to keep users and their Staging Posts on the same subnet.

So now we had addressed the performance issue, but move away from DMF somewhat. The issue now is, what do you do when your Staging Post disk is full? This is not a DMF system.

With some trepidation, we reintroduced the CIFS to user and NFS to LaRDS model, but now using the IS-3500 rather than the VM model.
We called them "Archive" shares.



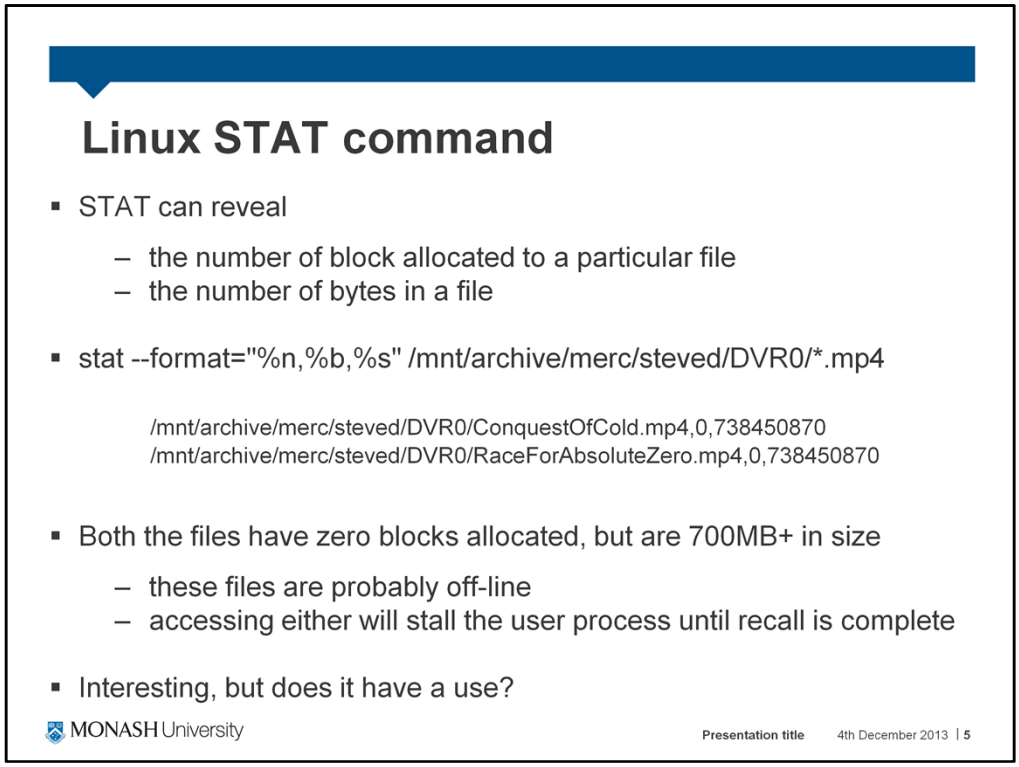
The Red Arrow is the client connection, reading and writing data content, mostly to the local disk.
The Blue Arrow represents the nightly RSYNC to LaRDS for the recovery of recently updated live data on the IS-3500 disk.
The White Arrow is the occasional file recovery for something lost within the last 24 hour cycle. (only used on three occasions).

The Green Arrow is the archive “passthru” share connection.

The “Archive” share has the characteristics of DMF.

Users mapped a drive or mounted a share on the IS-3500 Samba server using their AD credentials. Now they could see a second mount that allowed very large amounts of data to be stored.
It still does not have the ability to relay the on-line/off-line status of a file. The user desktop just stalls when a file needs to be recalled.
The DMF client tool set can be installed on the IS3500 samba server, but Samba does not have the smarts to relay this extended file information.

But it turns out there is a way to detect what files are on-line or off-line using the Linux STAT command.

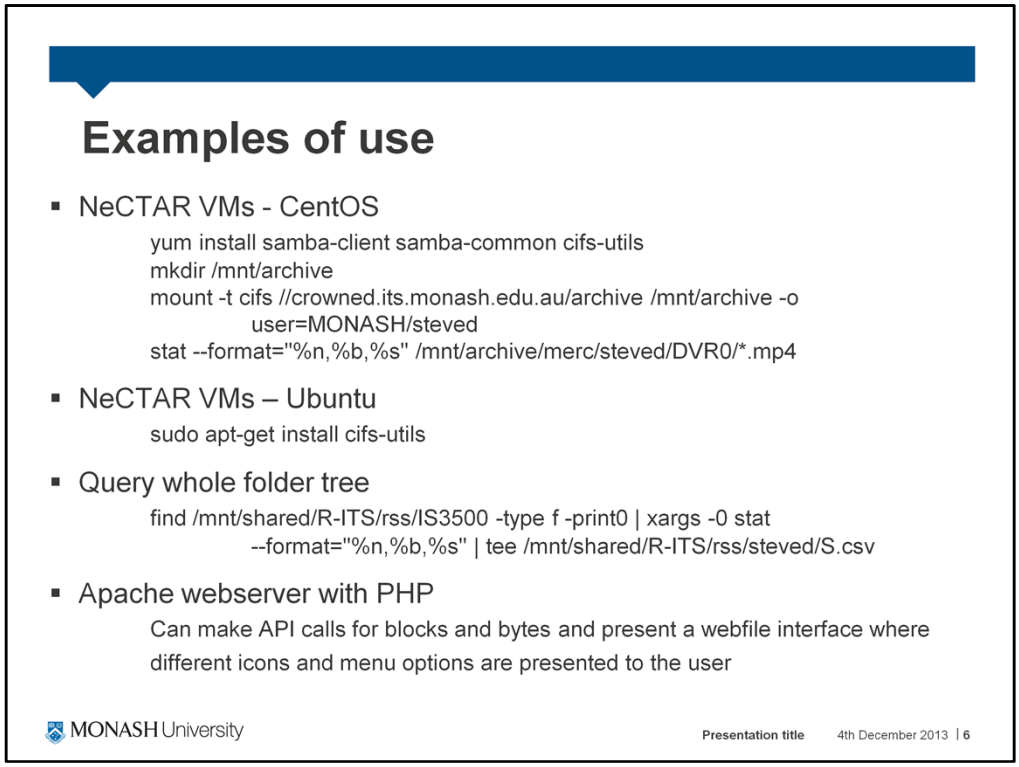


A few provisos have to be noted here…..

STAT is a command line utility of Linux, so you have to be on some kind of Linux based machine.
Not everyone is going to be allowed to SSH to the Samba server to run STAT.
Just about all will be on a desktop client with a CIFS share mounted.

The surprise is, that a Linux instance running a CIFS client that then mounts a CIFS share can still make use of the Linux STAT command and get exactly the same results as a SSH session logged in to the Samba server itself. So you can detect whether the number of blocks allocated are sufficient to contain the files size in bytes.

So what do you do with a DMF client that is not a DMF client?
Where might that be useful?

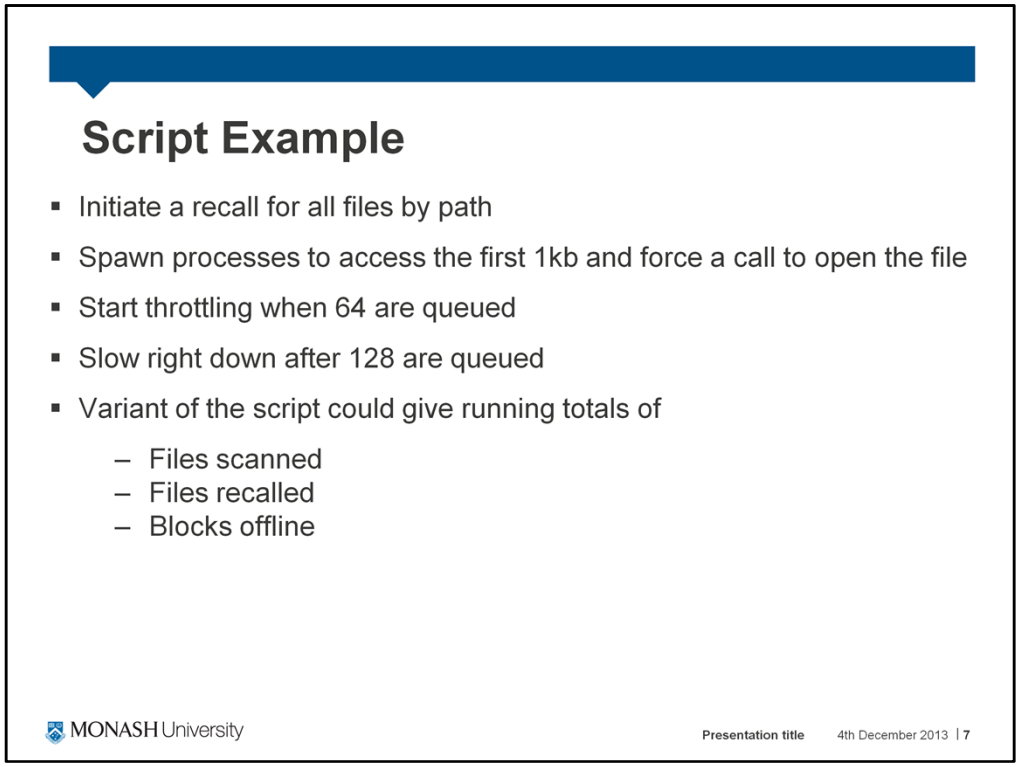


You might ask why this might be important for NeCTAR VMs.
This is where the plot thickens somewhat and RDSI gets a mention.

Most users of NeCTAR are used to doing block file things. Swift/CEPH on Openstack.
All of which works real fast for the compute cloud.
But they still need long term “Volume” storage while their image is just an image waiting to become a live instance with a new IP address.
NFS mounts, but that might require knowing the IP address of the instance would need updated permissions and IP-allows to get connected.

Not many RDSI nodes have a handle on this, or how to sort it out.

But, if you had a CIFS connection, you don’t care about the IP address, you only need the valid Posix credentials and the connection is made.
What's more, the STAT command example shows that Linux commands still work through that connection.




```
#!/bin/sh
#
# Philip.Chan@monash.edu
# November 2013


if [ "x$1" == "x" ]
then
  echo
  echo "DMF client-side file recall script"
  echo "Usage:"
  echo "    $0 <path>"
  echo
  echo "Best practice:"
  echo "  (a) cd to the directory with all the files to be recalled"
  echo "  (b) $0 . &"
  echo
  echo "  On the MCC, this script is never to be run on the login node"
  echo
  exit 1
fi


# retrieve a list of files on this path
find $1 -type f > /scratch/RCLIST.$$
```

```
for i in `cat /scratch/RCLIST.$$`
do
  # if non-empty file has 0 disk blocks --> assume to be on DMF
  DMFFLAG=`stat --format=%n\ %b\ %s $i | awk '{ print (!$2 && $3) ? 1 : 0 }'`


  if [ $DMFFLAG == "1" ]
  then
    # retrieve the first 1kb of the file and dump to limbo
    head -c 1k $i > /dev/null &
  fi

  # throttles to 64 active requests or thereabouts
  COUNT=`pgrep head | wc -l`
  if [ "x$COUNT" != "x" ]
  then
    if [ $COUNT -ge 64 ]
    then
      sleep 120
      if [ $COUNT -ge 128 ]; then sleep 120; fi
    fi
  fi
done


rm -f /scratch/RCLIST.$$
```

What needs to be done

- Please SGI
  - To me, the DMF interface is user unfriendly, locked in a time warp between the groovy command line of the 20th century and a futuristic LiveArc integrated streamlined environment of the 21st.
  - The linux DMF client kit needs to be an easily installable module for OpenStack, Ubuntu and CentOS. Make it NeCTAR friendly.
  - We as DMFers need to share the halfway hacks like STAT that can use any Linux/HSM combo.
  - A much simpler DMF client install kit needs to made available for Windows and Macintosh end users. (I wish I was SGI product manager for this deliverable. Users drive uptake and sales.)