DIVERSE Cluster Project ©2003

University Visualization and Animation Group
Virginia Tech Blacksburg, Virginia
by
Patrick Shinpaugh
Andrew Ray
Ron Kriz

Overview

The DIVERSE cluster project was initiated to realize a lower cost and more efficient CAVE system utilizing the DIVERSE API. At the Viriginia Tech CAVE, four walls displaying stereoscopic images provide an immersive environment using stereoscopic shutter goggles. Currently an SGI mainframe with a multi pipe graphics board drives the CAVE. The clustering solution is named DADS (DIVERSE Adaptable Display System).

LDAP (lightweight directory access protocol) was used to provide directory services and allow a common login for all machines in the cluster. The use of NFS (network file sharing) is also required to export directories to the cluster LDAP clients including common access to /home directories from the console/server machine. The specifics of implementing LDAP and NFS will not be discussed in this document. It is possible to share the install directories for DIVERSE (DTK and DPF) over the network and thereby have central installation directories, but it could result in network bandwidth limitations.

All documentation assumes the use of Redhat. If another distro is being used or UNIX the paths to commands may be different, or the distro may have another method to start/stop/restart processes/daemons.

CAVE Physical Configuration

The system is composed of 6 machines. The tracker server runs the is900 server process which processes the head tracker and wand tracker location as well as the wand joystick and buttons and stores the state in DTK network shared memory. The tracker server will not be discussed in this document. The console/server is used to run applications on the DADS system and is also the LDAP server for the DADS system. The remaining four machines are the graphics nodes of the system and each display one wall of the CAVE. The console and client machines each have a nVidia Quadro FX 3000G graphics card with framelock. At this time the console machine is not part of the framelock. The framelock master (one of the client graphics nodes) uses the 3 pin DIN output to send stereo signal to the stereo emitters. Genlock (house sync) is not used by the

DADS system, though it could easily be implemented if required.

The VGA along with mouse and keyboard output for each of the graphics nodes is connected to 4 Extron Electronics RGB 109xi interface boxes. The VGA signal is separated into RGB signal for output to an Extron Switcher. The VGA signal as well as the keyboard and mouse outputs are connected to a BlackBox Matrix ServSwitch KVM switch. This is used for maintenace when the monitor is not set to use the projector. The RGB cables from each of the Extron RGB Interfaces is routed through an Extron Electronics CrossPoint Matrix Switcher. Both the DADS system and the SGI box have 4 sets of RGB inputs into the switcher. There are four sets of output which provide the RGB video signal to the projectors.

Linux Installation and Configuration

For the current configuration RedHat Linux 8.0 was installed on the five machines; the tracker server machine will not be discussed in this document. A custom installation was chosen with X and KDE installed. All development options were chosen. In addition to the custom installation, glut, glut-devel, and openldap-clients and their dependencies were installed on the LDAP client machines. For the LDAP server Gtk-Perl, gtkglarea, and openldap-servers and their dependencies were also installed in addition to the client dependencies. The GTK modules are only necessary for the LDAP configuration utility. The monitor for the cluster PCs was set as unknown/unprobed since the video display may be connected through the KVM or projected on the walls of the CAVE. The specifics of implementing LDAP and NFS will not be discussed in this document.

The latest nVidia drivers which implement nvidia-settings should be installed. After following the driver installation instructions, there are important modifications that need to be made to the XF86Config. For framelock it is necessary to add a server flag option. To prevent blanking from occuring there are two server flags which need to be set. See the ServerFlags section of the XF86Config file in Appendix A.

The Monitor section of the XF86Config file should also be modifed. Because we have 2 montiors (a projector and a common display through KVM) for each client, there are 2 Monitor sections. However, only 1 monitor is active at any time. We simply switch between them by switching their numbers and restarting the X server. This is only done when maintenance is required on one of the client machines. The Monitor section configured for the projectors will probably require that modelines be specified in order to have the vertical refresh frequency desired. An Xfree86 modeline generator conforming to the GTF standard is available at http://gtf.sourceforge.net.

The cluster clients must boot into run level 3. To force the machine to boot into runlevel 3 instead of runlevel 5, edit the /etc/inittab file. Look for the line similar to

id:5:initdefault:

and replace the 5 with a 3. This can be postponed until all installations have been performed and GUI interaction is no longer necessary.

This is necessary because run level 5 will not allow a remote process to instantiate graphic applications. The .xinitrc run control script is used to accomplish this task. If it is necessary to run xinit manually, be sure to run xinit as the root superuser as a background process.

[root@machine somedir]# xinit &

Please see Appendix A for a listing of the .xinitrc file for the framelock master and the framelock clients.

One machine is chosen as the framelock master and all others will be slaves to that master machine. The framelock master must have permission to access the displays of all of the framelock clients. From the framelock master machine call

xhost +client

where client is the hostname of the client for each framelock client. Each framelock client must have access to the framelock master display. From each framelock client call

xhost +server

where server is the hostname of the framelock master machine.

Use the nvidia-settings tool to set up the machines for framelock. It should be configured from the framelock master machine. Within the nvidia-settings tool, click on the framelock tab from the list on the left. Click the Add X Screen button and add the framelock master host first. The format should be

dads1:0.0

or

dads1.your.domain:0.0

where dads1 is the hostname of the framelock master and your.domain is your specific domain. Each client should then be added similarly. If there is an error adding any of the machines, verify that xhost has been set up properly for the framelock master and each of the clients.

After all of the machines have been added, click on the Enable Framelock button. This enables framelock which will synchronize vblanking on all of the framelocked machines. Looking on the

back of the machines at the framelock connectors, the lights next to the cat 5 cables should be flashing. If they are not flashing, verify that you have completed each step properly. Also verify that the cat5 cables connected to the framelock ports are not also connected to any network ethernet ports (this could damage your graphics card).

DIVERSE

The cluster solution is built upon the DIVERSE APIs using both DTK and DPF for its implementation. DTK 2.3.1 (or greater) and DPF 2.3.1 (or greater) must be installed as well as any dependencies. Future versions of DIVERSE should also have DADS support. If you are looking to use Redhat 9.0 or Fedora 1.0, you should download the SGI Performer 3.1 version as previous versions of Performer do not function properly beyond Redhat 8.0. Installation and download instructions for both APIs can be found at http://diverse.sourceforge.net/archive/.

Implementation

Unlike other solutions we do not use MPI or other parallel API. Instead we used DTK shared memory segments and TCP sockets to communicate between cluster console/server and cluster clients. Using DSOs (dynamic shared objects or plugins) and daemons (background processes) in conjunction with graphics hardware capable of framelock and swap sync, we were able to implement a highly efficient and relatively low cost system to drive the CAVE walls with full synchronization.

A client-server system was implemented where each client runs a daemon to receive and interpret messages. However, the servers are either applications or DSOs used by DPF/DTK applications. All messaging is handled by DTK shared memory segments created specifically for communicating between server and clients. A starter DSO was implemented such that when it is loaded it will gather the necessary information to replicate the process and pass it to the clients.

DADS Installation

The cluster solution requires the installation of DTK 2.3.1 and DPF 2.3.1 and all of their dependencies. Download the DADS solution tarball from http://diverse.sourceforge.net. Extract the tarball and from the extracted dads directory run

```
make
```

which will build all of the daemons, programs, and DSOs. As root, install the binaries by running

```
make install
```

which will install the binaries in the dtk directory as determined by dtk-config. Daemons and programs are installed in the dtk/bin directory. DSOs are installed in the dtk/etc/dtk/augment_DSO directory.

The Makefile for the nvswapbarrier DSO will check to see if the graphics card on the current machine supports the OpenGL GLX_NV_swap_group extension. The nvswapbarrier DSO will only be built and/or installed if the graphics card supports the extension.

Environment Variables

There are several environment variables used with the cluster solution. Some are new and some are existing environment variables from DPF and DTK. All environment variables are set on the console/server machine and will be passed to the clients as needed.

DPF_DSO_FILES – From DPF. Use this in the console/server side to specify the DSOs to be loaded on the server.

DPF_CLUSTER_DSO_FILES – Specify the DSOs to be loaded by all clients.

DPF_CLUSER_DSO_FILES_host – Specify the DSOs to be loaded specifically for host where host is the base or alias of the hostname (i.e. The host for somehost.example.net would be somehost).

DPF_CLUSTER_ENV – Specify environment variables to be passed to all client machines. The environment variables must be set on the console/server machine prior to running diversifly.

DPF_CLUSTER_ENV_host – Specify environment variables to be passed to host where host is the base or alias of the hostname (i.e. The host for somehost.example.net would be somehost). The environment variables must be set on the console/server machine prior to running diversifly.

DPF_CLUSTER_CLIENTS – a colon separated list of hosts which should all be running the dListener daemon.

DPF_CLUSTER_SERVER – the host name of the console/server which should not be running dListener. This environment variable should be set to the hos name of the cosole/server machine.

DPF_CLUSTER_SYNC_SEGMENTS – Specify a colon separated list of DTK shared memory segments to be synchronized through navWrite and navRead. The shared memory segment values will be passed from the console/server machine using navWrite to all of the client machine through navRead.

Daemons

dListener

The daemon listens for messages on a shared memory segment. The messages are for starting a program, stopping a program, setting an environment variable, unsetting an environment variable, and checking the status of the listener daemon. The dListener must be running on all client machines or no actions will be taken on that client and the application will lock on the server and clients where it does spawn as the syncing requires communication from all clients.

dRepeat

This daemon will repeat shared memory segments on another named segment. It sleeps between reads and writes to reduce CPU usage and give other processes time.

```
dRepeat will repeat shared memory segments such that other machines
can connect via remote shared memory. There are 2 methods of
repeating the shared memory segments with this daemon. The first
is using --segment and providing a list of segments where the
repeated segments will each have the basename with a suffix of
repeat added to the ending. (i.e. dRepeat --segment head wand
will result in repeated shared memory segments of headrepeat
and wandrepeat).
The second method is to use --from segment --to segment pairs
where the from segment will be repeated with the to segment
name. Within each pair the from segment must precede the to
segment.

Options:
    --segment segmentlist  This will repeat each segment in the segment
    -s        segmentlist  list with a suffix of repeat (i.e. dRepeat
                           --segment head wand will result in repeated
                           shared memory segments of headrepeat and
                           wandrepeat).

    --from    segment      Must be paired with a --to segment. Specify
    -f        segment      the existing segment to be repeated.

    --to      segment      Must be paired with a --from segment. Specify
    -t        segment      the name of the repeated segment.

    -h | --help            Shows this usage/help information.
```

Applications

dListener-stop
This application

dKiller

This application is used to kill processes on client machines which are no longer functioning properly. The dListener daemon must be running on the client machine for this application to function properly.

```
dKiller is used to kill processes which were started by the dListener
daemon running on a client machine. Processes can be killed on a specific
machine or all machines defined in the DPF_CLUSTER_CLIENTS environment
variable. A specific program process can be killed or all processes
started by the dListener daemon can be killed.

Options:
    hostname          If [hostname] is defined then a message will be
                      sent to that client to kill the processes.
                      Otherwise the message is sent to all hosts
                      defined in the DPF_CLUSTER_CLIENTS environment
                      variable.

    --kill progname   Use the [kill program] option to kill a specific
    -k     progname   program which was started by the dListener where
                      program is the name of the process to be killed.

    --killall         Use the [killall] option to kill all processes
    -a                which were started by the dListener.

    -h | --help       Shows this usage/help information.
```

dSetEnv

This application is used to set environment variables on client machines which are running the dListener daemon.

dSetEnv provides a means to set environment variables using the dListener daemon running on a remote machine. The environment variable can be set for all clients listed in the DPF_CLUSTER_CLIENTS environment variable on the local machine or can be set using the computer/host name option for individual machines.

Options:

-c hostname These options will set the name of the client
--computer hostname as the hostname provided for which to set the
--hostname hostname environment variable.

-e name value This mandatory option specifies the environment
--env name value variable to be set with the specified value.

-h | --help Shows this usage/help information.

dUnsetEnv

This application is used to unset environment variables on client machines which are running the dListener daemon.

dUnsetEnv provides a means to unset environment variables using the dListener daemon running on a remote machine. The environment variable can be unset for all clients listed in the DPF_CLUSTER_CLIENTS environment variable on the local machine or can be unset using the computer/host name option for individual machines.

Options:

-c hostname These options will set the name of the client
--computer hostname as the hostname provided for which to unset the
--hostname hostname environment variable.

-e name This mandatory option specifies the environment
--env name variable to be unset.

-h | --help Shows this usage/help information.

dCreateShm

This application is used to create shared memory segments or messaging post offices. It can also be used to create the standard cave shared memory segments head, wand, joystick, and buttons.

dCreateShm is used to create shared memory or post office segments. Any combination of options is valid though it is the repsonibility of the user to avoid duplicating names.

Options:

--shm name size This will create a shared memory segment with
-s name size the specified name and size.

--po name This will create a PostOffice with the name
-p name specified for use with messaging.

--cave This will create the standard shared memory
-c segments head, wand, joystick, and buttons.

-h | --help Shows this usage/help information.

dExec

This application can be used to start applications remotely on a client running the dListener daemon.

dExec provides a means to initiate programs/processes using the
dListener daemon running on a remote machine. The program/process
can be started for all clients listed in the
DPF_CLUSTER_CLIENTS environment variable on the local machine
or can be set using the computer/host name option for individual
machines.

Options:

-c hostname These options will set the name of the client
--computer hostname as the hostname provided for which to start
--hostname hostname the program/process.

-p name args This mandatory option specifies the name of the
--program name args program/process to be started and its arguments
-h | --help Shows this uasage/help information.

soundPlayer

This is an application that will play a sound on a specified machine that is running a dListener daemon.

DSOs

starter

The starter DSO will connect to the message post offices of all clients defined in the DPF_CLUSTER_CLIENTS environment variable and send messages telling all clients running dListener to start the application. The message wille consist of the same command line arguments and will also pass the DPF_CLUSTER_DSO_FILES to the client as DPF_DSO_FILES. The starter DSO is required for the cluster solution to work properly. It is the method by which the client machines running dListener are told to start a program.

death

The death DSO is used to exit the client applications normally when the console/server application has been requested to exit normally. This should be included in the DPF_CLUSTER_DSO_FILES environment variable used on the client machine

navWrite

The navWrite DSO uses a software syncing solution to synchronize the location and head tracker position with clients running the navRead DSO. Must be used in conjunction with the navRead DSO or another navigation DSO implemented to synchronize similarly to the navRead DSO.

navRead

The navRead DSO uses a software syncing solution to retrieve the location and head tracker position with the console/server machine running the navWrite DSO. Must be used on the client in conjunction with a server running the navWrite DSO or another navigation DSO implemented to synchronize similarly to the navWrite DSO.

nvswapbarrier

This DSO provides swap barrier support for NVidia graphics cards supporting framelock and the NV_swap_group extension (i.e. NVidia Quadro FX 3000G).

dHideCursor

This DSO hides the cursor within the window associated with the application while it is running.

dPerformance

This DSO logs performance to a file called fps.log which it stores in the local /tmp directory. The file contains performance fps information for every 60 frames, and the totals when the

application is stopped.

vtCaveClusterGroup

This DSO provides all of the functionality necessary to run the DADS system from the console machine. You should modify the server and client host names to match your CAVE/RAVE system.

vtCaveClusterClientGroup

This DSO provides all of the functionality necessary for the clients to run properly in the cave with the exception of the display. Displays must be specified for each client. Includes xkeyboardMouseInput, caveDTKInput, setHeadView, caveSim, navRead, death, toggleObjectsGroup, toggleScreenFrame, and debugHeadsUp DSOs.

vtCaveDisplayFront

The standard display provides fullscreen 1280 x 1024 resolution at 0 degree offset from center with stereo support.

vtCaveDisplayFloor

This display provides fullscreen 1280 x 1024 resolution at -90 degree rotation about the x axis with stereo support.

vtCaveDisplayLeft

This display provides fullscreen 1280 x 1024 resolution at -90 degree rotation about the z axis with stereo support.

vtCaveDisplayRight

This display provides fullscreen 1280 x 1024 resolution at 90 degree rotation about the x axis with stereo support.

Example

The dtk-server daemon should be running on all machines being used in the DADS system which includes the console and the client machines. The dListener daemon should be running on the client machines only.

The following script will run diversifly sub.pfb on the console machine and 4 cluster clients named dads1, dads2, dads3, and dads4.

```
export DISPLAY=localhost:0.0
export DPF_DSO_FILES=desktopCaveEmulateGroup:starter:navWrite
export DPF_CLUSTER_SERVER=dadsconsole
export DPF_CLUSTER_CLIENTS=dads1:dads2:dads3:dads4
export DPF_CLUSTER_DSO_FILES=vtCaveClusterClientGroup:nvswapbarrier
export DPF_CLUSTER_DSO_FILES_dads1=vtCaveDisplayFront
export DPF_CLUSTER_DSO_FILES_dads2=vtCaveDisplayFloor
export DPF_CLUSTER_DSO_FILES_dads3=vtCaveDisplayRight
export DPF_CLUSTER_DSO_FILES_dads4=vtCaveDisplayLeft
diversifly sub.pfb
```

The script sets the DISPLAY to the localhost display screen 0. The DPF_DSO_FILES is used to set the DSOs to be loaded for the console/server machine (machine from which the script is run) to desktopCaveEmulateGroup (part of DPF), the starter DSO, and the navRead DSO. The name of the console/server machine is set using the DPF_CLUSTER_SERVER environment variable. This will be sent to each of the clients. The DPF_CLUSTER_CLIENTS environment variable is used to specify the client machines that will run the application. The DPF_CLUSTER_DSO_FILES environment variable is used to specify the DSOs which will be loaded for all of the client machines. The DPF_CLUSTER_DSO_FILES_dads1 environment variable is used to specify the DSOs which will be loaded specifically for the dads1 client machine. The same is true for each of the DPF_CLUSTER_DSO_FILES_dadsx environment variables which provides a means of specifying individual DSOs to be loaded. Finally the application diversifly is called.

When diversifly is called, the starter DSO will send a message to each of the client machines listed in the DPF_CLUSTER_CLIENTS environment variable which will include the command line arguments for the call to the application (i.e. diversifly sub.pfb). The paramaters of the DPF_CLUSTER_DSO_FILES environment variable will be combined with the parameters for the DPF_CLUSTER_DSO_FILES_dadsx environment variable for each client machine and renamed to DPF_DSO_FILES. This DPF_DSO_FILES environment variable will be sent as part of the message to each client (i.e. dads1 will receive a DPF_DSO_FILES environment variable as DPF_DSO_FILES=vtCaveClusterGroup:vtCaveDisplayFront). The message also includes the user id and the current working directory.

The dListener running on each of the client machines checks for messages on the COMMAND_SEGMENThost shared memory segment where host is the base name of the machine hostname. The dListener daemon will interpret the message and will attempt to start the program. First it will fork a process, set the new process as a process leader, set the user id of the process to match the userid from the message, change to the current working directory, set the DPF_DSO_FILES environment variable and finally it will exec the commandline arguments passed to it. If everything goes as expected, all client machines will be running the application, each with its associated DSOs.

Appendix A.

/root/.xinitrc (framelock master)

```
xhost +dads2
xhost +dads3
xhost +dads4
xterm -g 80x24+50+50 &
/usr/bin/nvidia-settings
```

/root/.xinitrc (framelock clients)

```
xhost +dads1
xterm -g 80x24+50+50 &
```

/etc/X11/XF86Config

```
# File generated by anaconda.

Section "ServerLayout"
        Identifier     "Anaconda Configured"
        Screen      0  "Screen0" 0 0
        InputDevice    "Mouse0" "CorePointer"
        InputDevice    "Mouse1" "SendCoreEvents"
        InputDevice    "Keyboard0" "CoreKeyboard"
EndSection

Section "Files"
# The location of the RGB database.  Note, this is the name of the
# file minus the extension (like ".txt" or ".db").  There is normally
# no need to change the default.
# Multiple FontPath entries are allowed (they are concatenated together)
# By default, Red Hat 6.0 and later now use a font server independent of
# the X server to render fonts.
        RgbPath      "/usr/X11R6/lib/X11/rgb"
        FontPath     "unix/:7100"
EndSection

Section "Module"
        Load  "dbe"
```

```
        Load  "extmod"
        Load  "fbdevhw"
        Load  "glx"
        Load  "record"
        Load  "freetype"
        Load  "type1"
EndSection

Section "ServerFlags"
        Option     "BlankTime" "0"
        Option     "NoPM" "true"
        Option     "AllowNonLocalXvidtune" "1"
EndSection

Section "InputDevice"
#       Option "AutoRepeat"  "500 5"
# when using XQUEUE, comment out the above line, and uncomment the
# following line
#       Option "Protocol"       "Xqueue"
# Specify which keyboard LEDs can be user-controlled (eg, with xset(1))
#       Option "Xleds"                  "1 2 3"
# To disable the XKEYBOARD extension, uncomment XkbDisable.
#       Option "XkbDisable"
# To customise the XKB settings to suit your keyboard, modify the
# lines below (which are the defaults).  For example, for a non-U.S.
# keyboard, you will probably want to use:
#       Option "XkbModel"    "pc102"
# If you have a US Microsoft Natural keyboard, you can use:
#       Option "XkbModel"    "microsoft"
#
# Then to change the language, change the Layout setting.
# For example, a german layout can be obtained with:
#       Option "XkbLayout"   "de"
# or:
#       Option "XkbLayout"   "de"
#       Option "XkbVariant"  "nodeadkeys"
#
# If you'd like to switch the positions of your capslock and
# control keys, use:
#       Option "XkbOptions" "ctrl:swapcaps"
        #Option          "XkbOptions" ""
        Identifier  "Keyboard0"
        Driver      "keyboard"
```

```
        Option     "XkbRules" "xfree86"
        Option     "XkbModel" "pc105"
        Option     "XkbLayout" "us"   #Option          "XkbVariant"  ""
EndSection

Section "InputDevice"
        Identifier  "Mouse0"
        Driver      "mouse"
        Option     "Protocol" "IMPS/2"
        Option     "Device" "/dev/psaux"
        Option     "ZAxisMapping" "4 5"
        Option     "Emulate3Buttons" "no"
EndSection

Section "InputDevice"
        Identifier  "Mouse1"
        Driver      "mouse"
        Option     "Device" "/dev/input/mice"
        Option     "Protocol" "IMPS/2"
        Option     "Emulate3Buttons" "no"
        Option     "ZAxisMapping" "4 5"
EndSection

Section "Monitor"
        Identifier   "Monitor0"
        VendorName   "Monitor Vendor"
        ModelName    "Projector"
        HorizSync    15.0 - 130.0
        VertRefresh  38.0 - 180.0
      Mode        "1280x1024"
            DotClock      192.54
            HTimings      1280 1368 1560 1864
            VTimings      1024 1032 1036 1075
            Flags        "Composite"
      EndMode
EndSection

Section "Monitor"
        Identifier   "Monitor1"
        VendorName   "Monitor Vendor"
        ModelName    "KVM Video Display"
        HorizSync    31.0 - 80.0
        VertRefresh  36.0 - 76.0
```

```
        Option     "dpms"
EndSection

Section "Device"
        # no known options
       #BusID
        Identifier  "NVIDIA Quadro 4 (generic)"
        Driver      "nvidia"
        VendorName  "NVIDIA Quadro 4 (generic)"
        BoardName   "NVIDIA Quadro 4 (generic)"
        Option     "Stereo" "3"
EndSection

Section "Screen"
        Identifier "Screen0"
        Device     "NVIDIA Quadro 4 (generic)"
        Monitor    "Monitor0"
        DefaultDepth     24
        SubSection "Display"
                Depth     16
                Modes    "1280x1024"
        EndSubSection
        SubSection "Display"
                Depth     24
                Modes    "1280x1024" "1280x960" "1152x864" "1024x768" "800x600"
"640x480"
        EndSubSection
EndSection

Section "DRI"
        Mode        0666
EndSection
```