A Prototype Cloud-Based Visualization System for Unidata Applications (Final Report for 2016-Unidata Equipment Awards)

Arthur Person and George Young
Department of Meteorology and Atmospheric Science
The Pennsylvania State University

Abstract

Graphical visualization is a critical tool used for meteorological study. Whereas graphical representations in years past were made with paper and ink, today they are produced with computers using advanced analysis and display workstation software. In the case of a classroom, the cost of installing and maintaining high-end graphical systems can be prohibitive, making widespread use of applications such as the Unidata IDV difficult to accomplish economically. This project overcomes this difficulty by developing a prototype system that concentrates high-end capabilities in “the cloud” and requires only average-speed networks to connect remote thin clients to the cloud servers. The benefits of this approach include more efficient use of resources, lower cost of hardware and management, and improved accessibility to the end user.

The Penn State Cloud-IDV Prototype

A prototype “cloud” visualization system was purchased with Unidata award funds configured as two hardware servers running the Apache Mesos distributed systems kernel. A Mesos master server was configured on a Silicon Mechanics R137.v6 with a Xeon E3-1270v5 (4 cores at 3.6 ghz), 64 GB of RAM and a dual Intel 480 GB S3510 SSD OS mirror. One Mesos agent was configured on a Silicon Mechanics R2504.v6 with dual Xeon E5-2650v4 (24 cores at 2.2 ghz per processor), 256 GB of RAM and a dual Intel 480 GB S3510 OS mirror. 3-D graphics capabilities were enabled with an NVIDIA Pascal Titan X graphics card (3072 cores, 12 GB RAM) installed in the Mesos agent machine. The Mesos agent chassis is based on a SuperMicro 7048GR-TR and is designed to hold and power up to four 250W graphics cards. Administration of the Mesos distributed kernel was through the Marathon container orchestration platform which was installed on the Mesos master.

Once the Mesos/Marathon cloud environment was functional, a Docker container was constructed to implement a stand-alone IDV instance accessible using a VNC client. Since 3-D graphics are very demanding on display bandwidth, a VNC optimized for 3-D graphical display, TurboVNC with VirtualGL, was chosen as the heart of the graphical display engine (see the diagram “In-Process GLX Forking with an X Proxy” at http://www.virtualgl.org/About/Background). These packages were installed on the Mesos agent where the graphics hardware was located. Functionally, the IDV instance produces its output on an X display created by TurboVNC (the X proxy) which is then made accessible on a VNC port in the range 5900-5999. TurboVNC hooks into VirtualGL to utilize the 3-D hardware acceleration transparently to the user. The TurboVNC VNC ports are made externally accessible on port 5900 to VNC clients through the use of HAProxy running on the Mesos master. For the VNC client to work well, it must be optimized for 3-D graphics. In our case, we used MobaXterm (which appears to implement a TurboVNC client) but others are known to work also.

The Dockerfile used to build this configuration is as follows:

```
FROM centos:7
```

```
RUN yum -y update
RUN yum -y group install "X Window System"
RUN yum -y install mesa-libGL
RUN yum -y install mesa-libGLU
RUN yum -y install perl bc

# Install nvidia driver
ADD software/NVIDIA-Linux-x86_64-375.26.run /tmp/NVIDIA-Linux-x86_64-375.26.run
RUN sh /tmp/NVIDIA-Linux-x86_64-375.26.run -a -N --ui=none --no-kernel-module
RUN rm /tmp/NVIDIA-Linux-x86_64-375.26.run

# Add "vgluser" to the system
ADD runfiles/passwdadd /tmp
ADD runfiles/groupadd /tmp
RUN cat /tmp/passwdadd >> /etc/passwd
RUN cat /tmp/groupadd >> /etc/group
RUN rm /tmp/passwdadd /tmp/groupadd

# Copy configuration files to /home/vgluser
RUN mkdir -p /home/vgluser/.vnc
ADD runfiles/turbovnc_passwd /home/vgluser/.vnc/passwd
ADD runfiles/startup /home/vgluser/startup
RUN chown -R vgluser /home/vgluser && chgrp -R vglusers /home/vgluser

# Add turbovnc security configuration file
ADD runfiles/turbovncserver-security.conf /etc/turbovncserver-security.conf
RUN chown root /etc/turbovncserver-security.conf && chgrp root /etc/turbovncserver-security.conf \
&& chmod 644 /etc/turbovncserver-security.conf

USER vgluser
ENV LD_LIBRARY_PATH /opt/libjpeg-turbo/lib64:$LD_LIBRARY_PATH
ENTRYPOINT ["/home/vgluser/startup"]
```

The above Dockerfile assumes that supporting files in the “runfiles” and “software” directories have already been created. It also assumes that working versions of VirtualGL, TurboVNC and IDV have been built on the Mesos agent host system as these binaries will be used from the host rather than being built into the Docker image itself.

The Docker build command was:

```
docker build -t nvidia-idv .
```

Run manually, the Docker command would be similar to the following:

```
docker run \
   --detach \
   --rm \
   --privileged \
   --network="host" \
   -v="/tmp/.X11-unix:/tmp/.X11-unix:rw" \
   -v /opt:/opt:ro \
   -v /usr/local/IDV_5.3:/usr/local/IDV_5.3:ro \
   -v /data/ldm:/data/ldm:ro \
   -v /data/ldm/idv_bundles:/data/ldm/idv_bundles:rw \
   nvidia-idv
```

The Docker container created above runs a startup script that starts TurboVNC and then runs the IDV in the X window created by this TurboVNC instance. No window manager is started, so window actions are strictly limited to the controls available within the IDV application. Once started, the IDV is accessible through a VNC port (mapped through HAProxy) using a VNC client, in our case, MobaXterm on port 5900. Using the Marathon platform, this Docker image was scaled out to 48 instances, one per processor on the system, supporting up to 48 users. The ability of the system to handle 48 users is dependent on the workloads, but for light loads, the system appears effective at handling most of those users (see Test Results below).

There are some important points to note about the above configuration. First, the graphics hardware is made available to the Docker instances by mounting /tmp/.X11-unix within each instance. The VirtualGL component accesses the hardware acceleration through a host-based permanent X client started on the Mesos agent on display 1 at boot time and requires the Docker instances to run in privileged mode to gain access to the hardware acceleration. The IDV accesses the X11 display created by TurboVNC from the host's /tmp/.X11-unix location as well. Second, each instance must have access to the Mesos agent host network since externally accessible VNC ports in the range 5902-59xx will be created at random each time an instance starts (TurboVNC assigns a unique, unused, VNC port number for each IDV instance numbered in the 5900's which also corresponds to it's X11 port number). Lastly, the Docker instances make use of TurboVNC and IDV binaries built on the Mesos agent host rather than ones built within the instance itself in order to allow for clean builds (it proved difficult to build these packages within the Docker instance). For this to work, the NVIDIA driver must be installed in the Docker image and match the version installed on the host. Although these caveats break some of the rules of Docker philosophy for a transportable image, it still provides a mechanism for packaging replicable instances that can be started manually or in Marathon for as many users as the system will support.

Test Results

A natural question to ask is "how well does it perform?" Two tests were run to help answer this question. The first test utilized the glxspheres64 application that came with the VirtualGL package. This was packaged in a Docker image and started with Docker manually on the graphics server successively until it became unusable. The following table shows utilization versus number of instances running:

Number Of Instances	GPU Utilization Percent	Frames per Second
1	20	100
2	30	100
3	40	90
4	50	80
5	55	75
6	65	75

7	75	65
8	75	60
9	80	60
10	95	60
11	95	60
12	98	55
13	100	50
14	100	50
15	Jammed	

At about 12 or more instances, performance was noticeably degrading. Notably, however, although the 15th instance jammed when started, all the other instances continued to run.

The second test performed was a live test of IDV instances running through Marathon. An IDV bundle was created for a 3-D surface of 90 % relative humidity shaded by height for the 20170505 12Z gfs grid 211 data for 41 forecast hours. Forty-eight IDV instances were started in Marathon and then successive instances were accessed with a VNC browser and loaded with the bundle. The view was tilted slightly and then started into autorotation. GPU utilization versus number of running instances were recorded with the following results:

Number Of Instances	GPU Utilization Percent
1	8
5	20
10	35
15	29
20	42
45	98
46	100

Oddly, the utilization appears to drop between 10 and 15 instances, but this is due to the performance setting on the graphics card automatically adjusting from P5 to P0 meaning maximum power was being used (P0) after the 10th instance. Memory utilization on the graphics card was very low throughout the test. With GPU capacity saturating at 46 instances, it would appear that 25-30 instances in a classroom setting may work well as long as the 3-D graphics imagery is not too complex or rotated too intensively.

Several issues were noted during testing. Thin clients in the classroom lab exhibited a flickering screen during IDV option selection, but an office PC did not. This will require further investigation. Some tweaks may be necessary to the IDV to make it more functional in a non-window-manager system environment. For example, without a window manager, the dashboard and map view windows overlap each other. Fortunately, the “File” tab is accessible in both cases allowing one to switch between each window easily. On the other hand, clicking on “Help->User’s Guide” covers the control drop-downs and requires exiting and restarting the IDV to recover. Finally, since the Docker IDV instance runs as a generic user and has limited access to any underlying file systems, it is difficult to both save individual user context as well as access user datasets. This could be resolved by configuring the instance to allow an actual logon to a user account, but would also introduce potential security issues that would require careful consideration.

Conclusion

The Penn State Cloud-IDV prototype has shown that the potential for enabling many users to access the IDV or other visually demanding applications through thin clients in a cloud environment looks very promising. While the above configuration is a bit cumbersome, these deficiencies should gradually disappear as cloud technology becomes more adaptable to accelerated graphics hardware environments. The possibility of making accelerated 3-D graphics and the IDV ubiquitous in the classroom and lab environment is very exciting, and we will be exploring trial implementations of this prototype system at Penn State in the near future.