Part 3: Chromium in Practice

Chromium Configuration Tool (1)

A graphical “click and drag” interface for creating Chromium's Python config files.

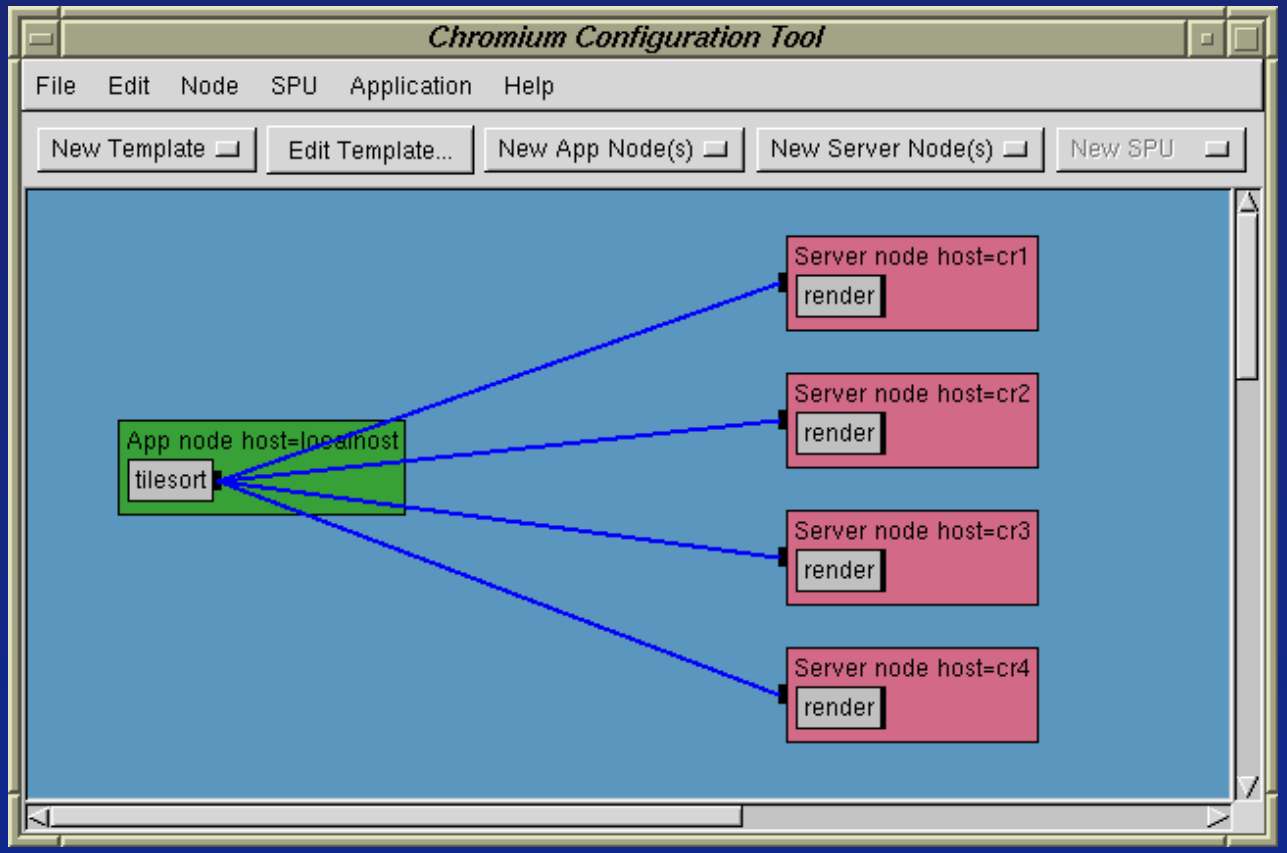
Written in Python, using wxPython GUI toolkit, so it runs on Unix and Windows.

Built-in templates for a few common types of configurations, such as tilesort and sort-last.

Click on nodes, SPUs to select them. Then use menu to set options, wire the components together, etc.

Chromium Configuration Tool (2)

Screenshot: Simple tile-sort config.



(demo)

Cr in the Production Environment

Concerns:

- Ease of use
- Desktop integration / DMX
- Correctness
- Miscellaneous
- Troubleshooting

Ease of Use (1)

Using an application with Chromium shouldn't be any harder than using an ordinary OpenGL application. For example, run 'visit' and have it automatically show up on a DMX/Tilesort display.

Once things are properly set up, that'll work.

But, it takes someone with some understanding of Chromium to get things in place.

Things you should do...

Ease of Use (2)

- Install Chromium on a shared filesystem or replicate it on all nodes (in same directory).
- crserver, crappfaker, etc should be the user's default search path.
- Chromium libraries should be in LD_LIBRARY_PATH or in etc/ld.so.conf
- Cr config files should be in a standard place.
- User should have permission to start processes on cluster nodes with rsh or ssh without typing in passwords.
- Cr config file should be set up to use Auto-start (more...)
- Auto config file selection should be set up (more...)
- Misc: DISPLAY env vars set correctly by default, xhost not interfering, etc.

Ease of Use (3)

Create a set of standard Cr configuration scripts for your site:

- Sort-first mural display
- Sort-last compositing
- Etc.

End-users shouldn't have to develop/modify the Cr configuration scripts (unless their advanced users).

Start with the Cr demo scripts and build from there.

What sort of things should be done in configuration scripts? (next slides)

Ease of Use (4)

Auto-start:

- Automatically spawn/run all the crserver and crappfaker processes on the appropriate nodes.
- node.AutoStart() method allows you to specify a command-line instruction, such as 'ssh <hostname> <command>' which the mothership will execute:

```
net_node = CRNetworkNode('mars')
net_node.AutoStart(['ssh', 'mars', 'crserver'])
```

Your environment should already be set up for password-less ssh or rsh execution, of course.

Ease of Use (5)

Auto-mothership configurations:

- Make symlink from libGL.so[.1] to libcrfaker.so so app uses Chromium's libGL replacement.
- When libcrfaker.so discovers there's no running mothership, a mothership config will automatically be chosen and initiated.
- The ~/.crconfigs file will map the application name to a mothership config:

```
visit /opt/Chromium/configs/visit.conf %m %p
*     /opt/Chromium/configs/default.conf %m %p
```

%p will be replaced by argv[0], %d replaced by current directory path.

Ease of Use (6)

Cluster configuration information:

- The ~/.crsite file describes the cluster: hostnames, display sizes, etc.
- Used by the graphical configuration tool (later) and configuration files so that such information isn't hard-coded in every configuration file.

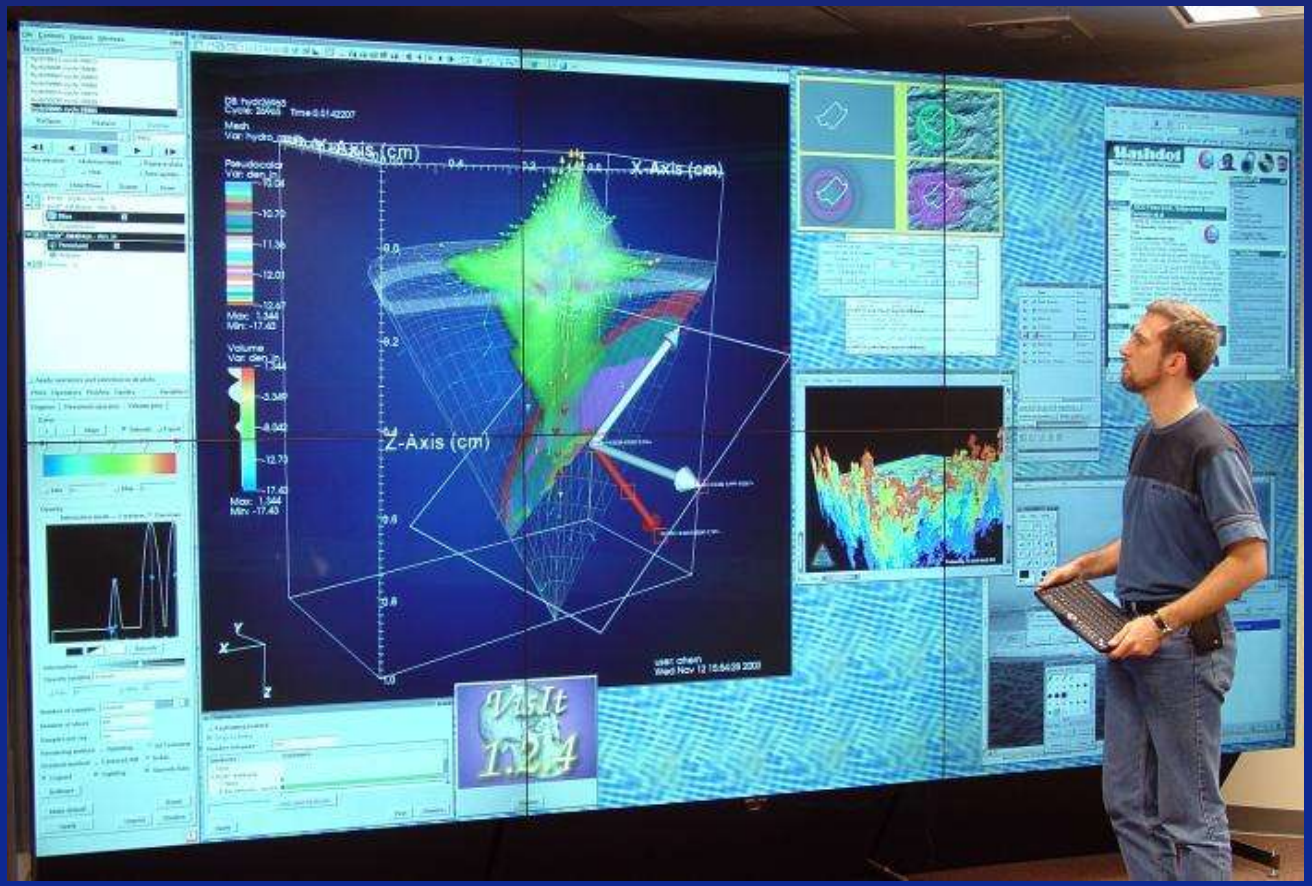
Miscellaneous:

- If the cluster has a batch allocation or session manager, that'll have to be solved on-site.
- The Python config file allows lots of flexibility there.

DMX

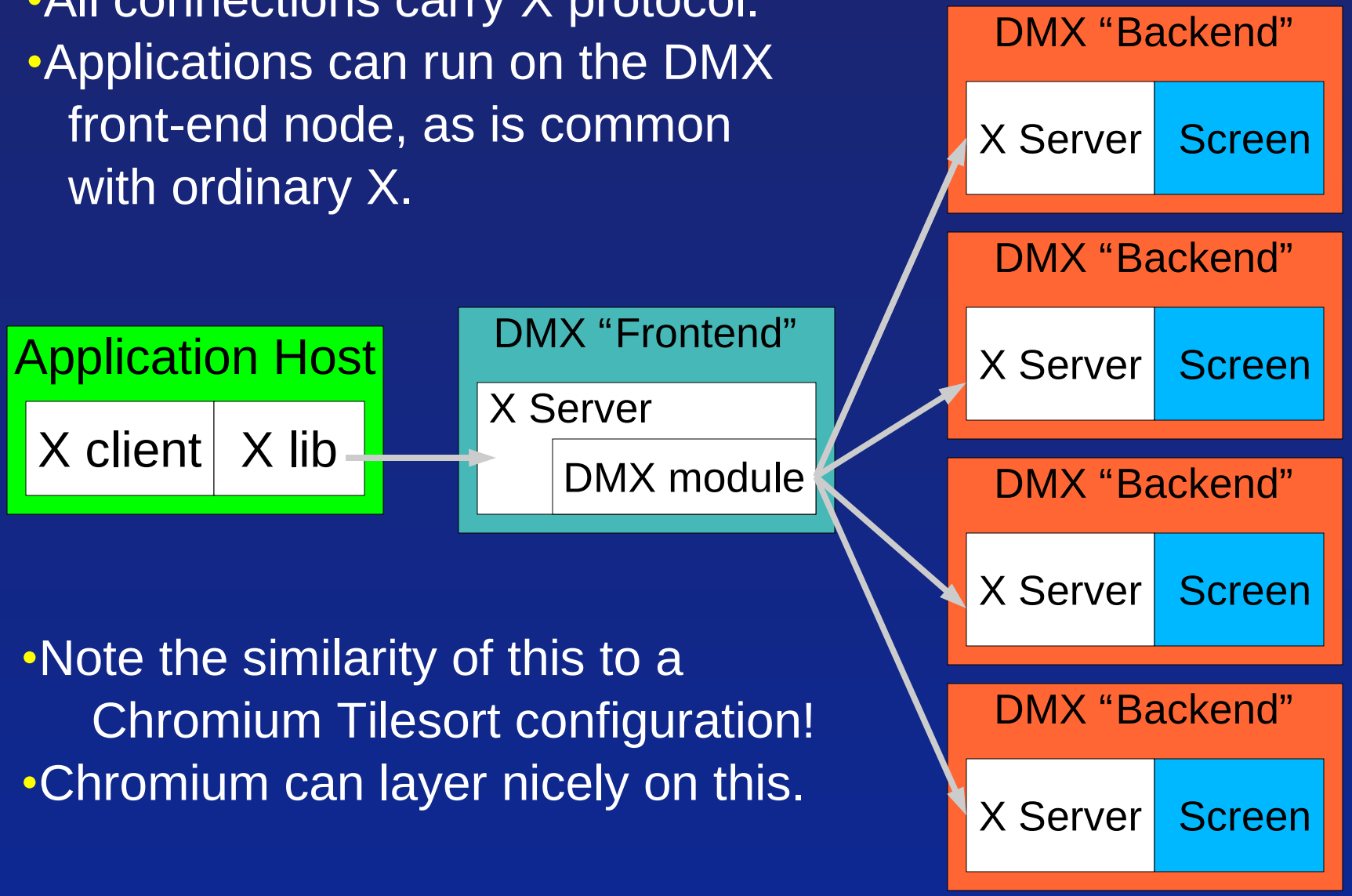
DMX = Distributed, Multi-head X.

- Allows an X desktop to span many screens on many machines.



DMX Diagram

- All connections carry X protocol.
- Applications can run on the DMX front-end node, as is common with ordinary X.



- Note the similarity of this to a Chromium Tilesort configuration!
- Chromium can layer nicely on this.

DMX + Chromium

- Chromium is DMX-aware
- DMX + Chromium = hardware-accelerated 3D integrated with 2D windows on a multi-screen desktop.
- Multiple window support. 3D rendering appears in the application window, not in a separate place.
- Resize/move work as expected. The Tilesort SPU watches for window changes and adjusts its tiling accordingly.

- The ideal way to run things – fewer rendering glitches because there's no “viewport tricks”.

Correctness

- Correctness w.r.t. OpenGL improves with each release.
- However, in some cases, it's nearly impossible to be 100% conformant with OpenGL.
- New Chromium tests exist to ferret out bugs (packer/unpacker tests, for example).
- There's a new test plan to run before Cr releases.
- Reports of correctness bugs seem to be on the decline.
- Rendering issues should be reported to the bug database or mailing lists so they can be fixed.

Miscellaneous

Robustness

- Has been improving with each Cr release
- More and more apps work “out of the box”
- Incorporate user feedback (better clean-up on exit, fix GL driver-specific glitches)

Site-specific:

- Some things are out of Chromium's control, such as permissions, session management, etc.
- Don't hard-code hostnames, port numbers, etc. into config files.

Troubleshooting

General categories:

- Rendering errors and glitches
- Crashes
- Poor performance
- Configuration errors

Troubleshooting: Rendering Errors (1)

Typical problem: rendering error with Tilesort SPU.

The tilesort SPU is pretty complicated and we still occasionally find bugs in it.

Things to try:

- Set `bucket_mode` to 'Broadcast'
- Set `dlist_state_tracking` to 0.
- Set `lazy_send_dlists` to 0.
- Increase MTU size
- Disable `optimize_bucket` on crserver.
- Report the bug!

Troubleshooting: Rendering Errors (2)

Is my parallel application correctly synchronized?

Troubleshooting: Crashes

Less common nowadays.

Things to try:

- Increase MTU size (if fixed, indicates a bug in the packer/unpacker code)
- Run another program and see what happens.
- Try smaller dataset
- Try fewer tilesort screens
- Using an old NVIDIA driver? Upgrade.
- Report the bug!

Troubleshooting: Poor Performance

Take a step back and consider what you're doing.

Tilesort: what kinds of commands (and how many) am I sending over the network? Lots of vertex or image data? Consider display lists.

Sort-last: how many pixels per image? How much bandwidth do I have?

Run 'top' on all nodes and see what's going on.

Test network bandwidth with other programs to know your limits.

Compiled with optimizations?

Got the latest ATI driver? glReadPixels is better now.

Troubleshooting: Configuration Issues

Does Python report an error? Usually easy to fix.

Does my config file make sense? Try loading it into the configuration tool to view the components graphically. Is everything connected?

Are my hostnames and file paths set correctly?

Am I forgetting to configure something entirely, like the tiling information?

Is the default DISPLAY env var set correctly? If not, you may be using (slow) indirect GLX rendering.

Troubleshooting: Misc

Are there any stale processes from previous runs hanging around? Maybe add 'killall' commands to Python script. When using GM it's actually hard for Chromium to detect termination.

Are multiple users trying to use the same default port numbers?

Does the app need Selection/Feedback? Use the Feedback SPU.

Does the app use vertex arrays? Use the Array SPU.

Take a look at the Chromium FAQ.

Support

When something goes wrong, who do you call?

Chromium is an open-source project.

No single person/party responsible for it.

Developers, such as myself, help out on the mailing list as time permits

Companies, such as TG, can be engaged for larger scale issues (new features, supporting new apps).

Chromium and Sci-Vis Applications

Why use Chromium for Sci-Vis? Primarily:

- Run unmodified applications on a large, multi-screen display wall. Ideally, with DMX too.
- Use Chromium's parallel rendering API extensions to insulate the application from details of the underlying rendering cluster. (different compositors, etc). You don't have to implement the compositing infrastructure in your application.

Chromium Vis Applications

Chromium is often used with:

- VisIt
- Paraview / vtk applications
- CEI Ensight/Enliten
- OpenDX – general sci vis
- VMD – molecular model viewer
- Vis5D – atmospheric visualization
- Catia - CAD
- NPB – large image viewer
- Raptor – written for Chromium
- OpenRM applications
- Inventor applications

Chromium Vis Concerns

Performance is a primary concern.

Tilesort: we're typically limited by the network. Recall, sending bulk vertex data isn't too efficient. Consider where display lists and textures live.

Sort-last: Only a few applications parallelized for Chromium at this point. Parallel apps need some modifications to use Chromium effectively.

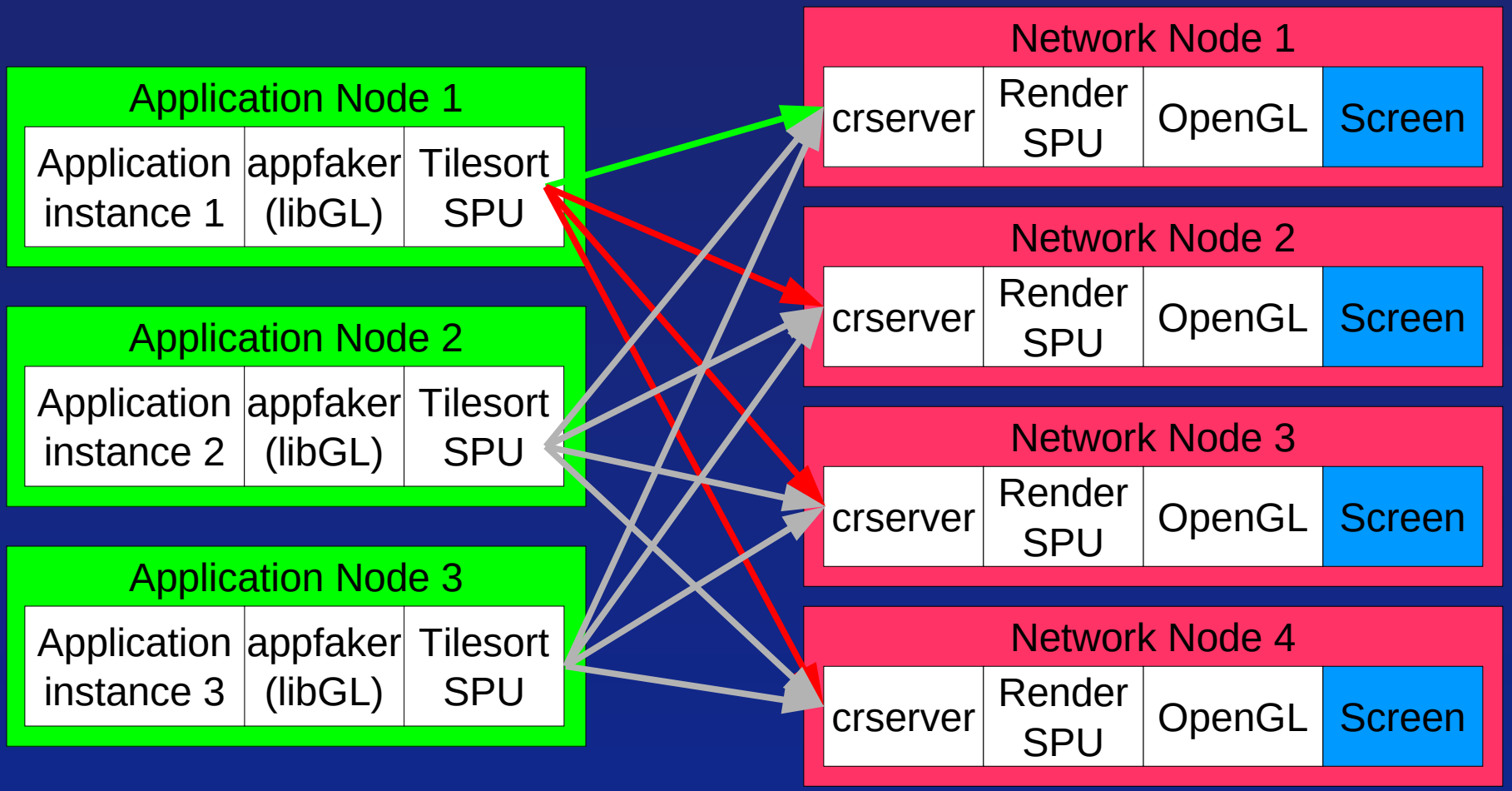
In both cases, optimizations like providing bounding boxes can help a bit.

Correctness and usability have improved a lot over the past couple years.

Chromium Ongoing/Future Projects (1)

- More OpenGL features, like 2.0 support
- Improved networking (Infiniband, MPI, multicast/broadcasting)
- Threaded networking to avoid blocking, both on client side (tilesort) and server side (network nodes). Diagram...

The Parallel Tilesort Problem



- Tilesort SPU send buffers to network nodes in round-robin order.
- Crserver processing incoming connections in round-robin order too.
- Can lead to lots of blocking – hence, threading would be good.

Chromium Ongoing/Future Projects (2)

- Threading should also allow more opportunity to overlap computation, networking and graphics.
- Optimizations in all areas
- New SPUs (new compositing, rendering)
- Hardware-synced tiled displays (sync the SwapBuffers and Stereo display)
- New rendering ideas:
  - ICE-T -like rendering (with DMX support)

The end.

Any questions?