Containers @ NCAR

[HPC]

Ben Matthews
matthews@ucar.edu

[CISL/Supercomputing Services Group]

CISL Computational & Information Systems Laboratory

NCAR

About the Speaker

- HPC SysAdmin for NCAR
- Occasional HPC user (CFD more than Climate)
  - While I'm going to pick on the Climate apps, I probably don't know what I'm talking about
- Main developer of Inception, the container solution in use on NCAR's HPC systems (at least for now)

Goals

- Explain Containers
- Expose idiosyncrasies of NCAR's environment with regard to containers
- Discourage people from demanding Docker/Singularity/The new Shiny :-)
- Explain our current setup
- Provoke discussion on how to improve

Anti-Goals/Disclaimer

- Political Correctness
- I don't intend to pick on any of our model developers or user support folks
  - Even if it may seem like I do
  - (and it is kind of fun)
  - (I mean.. WTF? https://github.com/wrf-model/WTF)
- **Some of the opinions here are mine alone - I don't necessarily buy into the hype**

What's a container?

- I don't know
- Linux has namespaces and cgroups
- *BSD has Jails
- Solaris has Zones
- **Not Magic**, Not new, and Not that hard
  - Sorry Docker :-)

```
matthews@laramie1:~> unshare -rm
laramie1:~ # whoami
root
laramie1:~ #
```

What's a container?

- A marketing construct?

Trademarks property of their owners.. yada yada. See https://www.docker.com/ https://singularity.lbl.gov/

No, Really, What's a Linux container?

- Namespace certain parts of the kernel per-process or per-process tree
  - Mount table
  - Process table
  - Network stack
- Put those processes in cgroups to restrict their resource usage
  - [well run] HPC systems have been doing this for years

HPC @ NCAR

- Mostly high-throughput, but some real HPC (20k-200k jobs/day)
- Lots of active users/projects
  - Many (most?) of them are geoscience focused - not CompSci

```
matthews@cheyenne1:~> getent passwd | wc -l
3219
matthews@cheyenne1:~> getent group | wc -l
1357
```

- Main system is 4032 Broadwell nodes + EDR [SLES12SP1]
- Smaller, Heterogenous Viz system [RHEL6.4]
  - Used for I/O heavy workloads as much as actual Viz
- Various even smaller test and research systems
- Much of the work is a few common packages, but there's a long tail of research apps.

HPC @ NCAR

- Stateless nodes (mostly)
- 32-128G of RAM/node
- Single GPFS instance available everywhere
- SLA governing how many nodes need to be up
- Users run what they like, within the constraints of their allocation
- Several in-house apps that evolve with the computing environment

HPC Culture at NCAR

- Make this easy for the user… by any means necessary
- It has to just work on our system, even if the apps will never work anywhere else
- Climate models have evolved around this philosophy
  - Lots of “porting” effort, even between practically identical systems

HPC Culture at NCAR: Magic!

- Compiler wrappers automagically find common libraries:

```
matthews@cheyenne1:~> printf '#include <netcdf.h>\nint main(int a,
char** v) {nc_open("test.netcdf", NC_WRITE,  NULL); return(0);}' > test.c
matthews@cheyenne1:~> icc test.c
matthews@cheyenne1:~>
```

- Batch scheduler? What's a batch scheduler? I just want to run!

```
matthews@cheyenne1:~> execdav
mem =
amount of memory is default
not setting x forwarding
Submitting interactive job to slurm using account sssg0001 ...

submit  cmd is
salloc  -N 1  -n 1 -t 6:00:00 -p dav --account=sssg0001 srun --pty --export=HOME=/glade/u/home/
matthews,PATH=/bin:/usr/bin,TERM=xterm-256color,SHELL=/bin/bash /bin/bash -l
salloc: Pending job allocation 332755
salloc: job 332755 queued and waiting for resources
salloc: job 332755 has been allocated resources
salloc: Granted job allocation 332755
salloc: Waiting for resource configuration
salloc: Nodes caldera03 are ready for job
Restoring modules to system default
bash-4.1$
```

Tight Coupling between HPC stack and Models

- Popular Model(s?) interact with the HPC stack directly
  - Lmod to find libraries
  - HPSS for archiving
  - PBS/Slurm/LSF for process management
    - Each model run may be many "jobs"

CISL Computational & Information Systems Laboratory
NCAR

What's all this have to do with Containers?

- Containers let (force?) the user to be the SysAdmin
- Most just want to run their models without having to think
- Models expect a complete traditional HPC stack
- **How can we provide this ~~crazy~~ rich environment in a container?**

Use-Cases

[User Facing]

- Education/Outreach
  - Help outsiders run the models, even if only small (single node?) scale
- "Cloud"
- Reproducibility
- Installing complicated/poorly designed software
- Collaboration with outside groups
- "sudo apt-get install"
- **New and Shiny!**

Use-Cases

[Systems]

- Containing bad behavior
  - **Cron**
- Staging upgrades
- Enterprise style uses (containing servers/lightweight VMs)

CISL Computational & Information Systems Laboratory

NCAR

Problems

- Licensing
  - Hard dependancies on commercial software (Intel/PGI Compilers, SGI MPI, SuSE, etc)
- Default image is *huge*

```
cheyenne1:~ # du -shc /glade/u/apps/ch
141G    /glade/u/apps/ch
141G    total
```

- Dependancies on terabytes of static model data
- **Anyone know what the size limit is on DockerHub? SingularityHub?**

Problems/Requirements

- Container runtime must not use system facilities that aren't cleaned up between runs, even if the job misbehaves or is oom-killed
- Container runtime must not give the user permissions they wouldn't otherwise have

Problems/Requirements

- Container runtime must not DDoS the internet (github?)
  - Or the various distro mirrors
    - oops, we've already done that
- Compute nodes don't even have internet
  - And the login nodes are already overwhelmed with compiles/ downloads

Problems/Requirements

- Any container commands must work first time, every time
  - Otherwise people get confused
  - Hm.. downloading everything from the internet might not be good then?
- HPC software is notoriously fragile
  - So, updating the distro every build isn't a good idea then? Docker….?
    - [but security?]

Problems/Requirements

- Users have asked for full featured (non-trival) containerized environments
  - X11
  - VNC
  - OpenGL
  - MPI/InfiniBand
- While maintaining portability

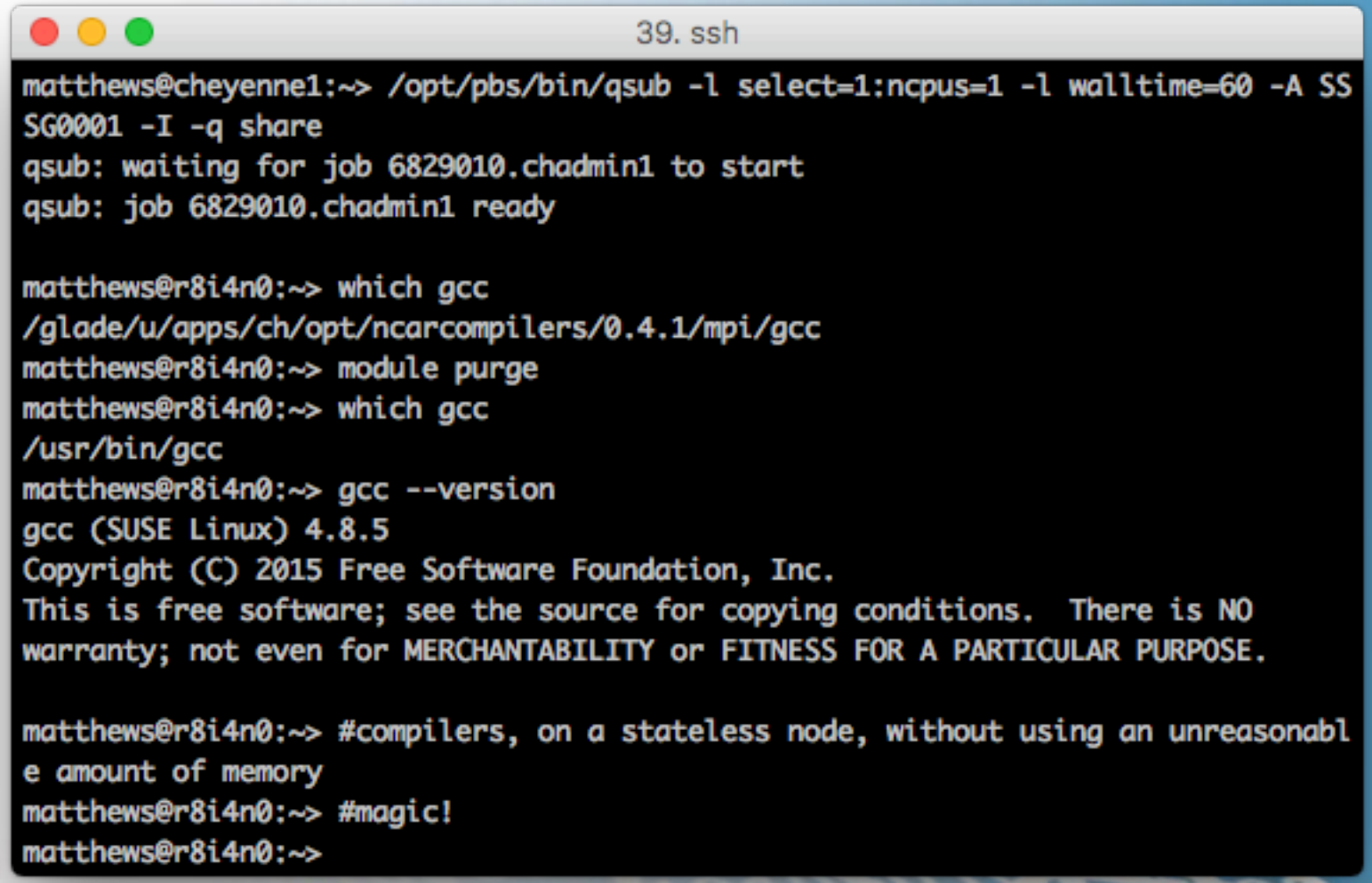
Problems/Requirements

- **Easy to use**
  - Even for people who don't care to learn about computing
- Safe for the system
  - No matter what people try to run
  - And we have enough users, someone will try everything

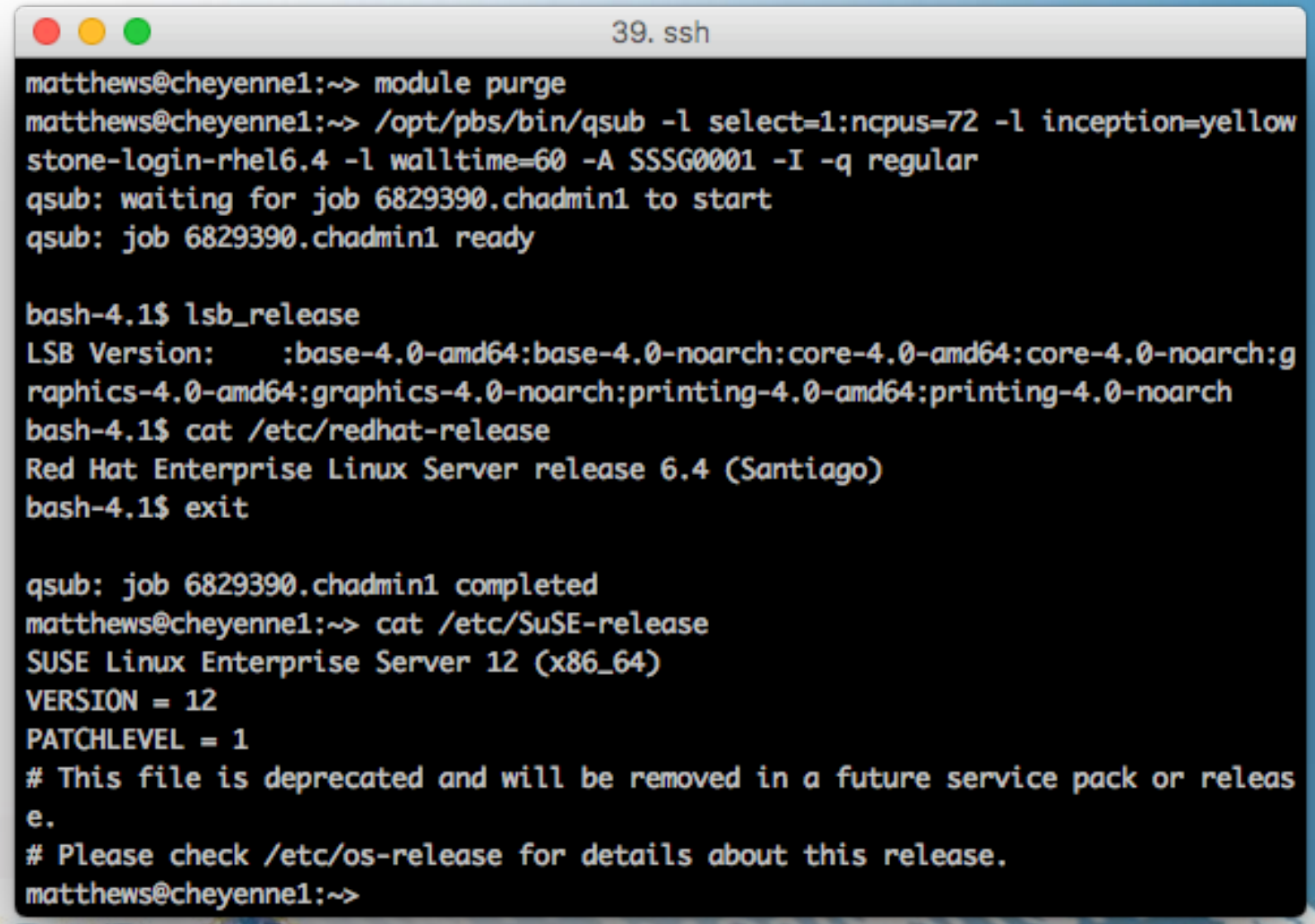
So, what do we have today?

- Docker on some infrastructure machines [no end users]
- Curated containers for HPC users
- Lightweight (developed in-house) runtime, tightly integrated with the scheduler
- Various containerized services
  - User might not even know

Containers Today: Transparency



Reproducibility
Yesterday's Bugs Today



That's cool, but how do I setup my own?

- You don't
  - For now…
- Build your environment on your own hardware
  - Or SingularityHub
- SysAdmins will make it available
  - After installing drivers/libraries as needed

CISL Computational & Information Systems Laboratory
NCAR

But "sudo apt-get"?

- Any day now…
- Supporting local container builds isn't hard, given a new enough kernel
  - Except that it still isn't really safe [yet?]
- I'm tired of rebooting broken nodes
- Vendors aren't that helpful when you find obscure kernel bugs
  - "Doing X breaks the kernel? So don't do that"

So, I heard Singularity is Secure

- Yeah, maybe, but the kernel is still broken
- Even if there were no “security” problems, there are still ways to break the kernel

… For Example?

- Force slab allocations outside your cgroup
- Mixed software versions confusing hardware
  - Shouldn't be possible, but…
    - InfiniBand
    - GPU
- Leak Resources
  - Loopback devices maybe? Singularity?
- Misbehaving Software that looks at only getuid()
  - [example removed to protect the guilty]

But the benefits are worth the risks, right?

- Well, it depends
- Let's review our use-cases

Container "Benefit": Reproducibility

- Sure, you can use old libraries on a newer system
  - … if the CPU is the same (or compatible)
  - … if the Interconnect is compatible
  - … if there aren't any bugs introduced by the new environment
  - … security?
  - … performance?

Container "Benefit": Reproducibility

- Does it count as "reproduced" if the system performance is drastically different?
  - Is your algorithm deterministic?
- Even on the exact same system, did you account for changes in performance due to age/temperature/uCode/etc?
- Did you account for the different set of Kernel bugs between runs?

Container benefit: Cloud/Compatibility with outside collaborators

- Again, containers don't really make code more portable
  - Does the cloud site have the same
    - CPU?
    - Interconnect?
    - Kernel?
    - Container runtime?
- Might work, might not

Container Benefit: Ease of installing complicated software/distribution to the public

- Ok, containers kind of help with this
- but… Fix your build system (and docs)
  - Really, it's not that hard
  - This "porting" concept between functionally identical x86 systems is ridiculous
  - Even porting between Linux systems with different CPUs really shouldn't be a thing
- ./configure; make; mpirun ./a.out
- If we don't **have** to make software portable, it's going to get even harder to move around. Do you really want to be locked into the Intel universe?

Container Benefit: "sudo apt-get install"

- Yeah, if we could do this safely, it'd be nice, sometimes
- But based on the logs, most people are just blindly following some tutorial when they try to do this
- So, maybe it wouldn't really help that much
  - (it's good to understand what you're doing)
- Again, fix your build systems

Container Benefit: "sudo apt-get install"

- Also, this is solvable other ways
- Anaconda
- Ports
- HPCInstall [an NCAR product]
- Spack
- It would be nice if these were a little more mature
- **Even with root, I don't install software as root - you never know what the installer is going to do**

Wow, you sound really negative toward containers

- Well, they're good for.. something.. um.. look, a shiny new thing!
- Actually, there are legitimate uses on the infrastructure side
  - Rebuilding the whole OS for every change might make sense
    - Always up-to-date
    - Admins can stage things and fix breakage

Conclusions

- Containers are just another shiny oversold thing that's been around forever
- There are uses, but for the most part, there are better ways
- Linux's container support isn't quite ready for prime-time
- Please stop demanding {docker, singularity, shifter} just to "try them out" without a clear use-case
  - If you just want to play with the new shiny thing, use your own hardware
    - Docker is fantastic on OSX and Windows

Why am I wrong?

- How can we make this better?
- Other questions?