nfsroot

overview and discussion points

Jim Garlick

garlick@llnl.gov

Livermore Computing

Lawrence Livermore National Laboratory

LLNL-PRES-615396

Presentation Topics

- overview
- diskless boot sequence walkthrough
- scalability questions for large clusters
- distributed network block device

overview

nfsroot scope

nfsroot makes a root fs image sharable via NFS

The main unique thing about nfsroot is it is contained in the diskless root image. There is no server component, therefore configuring DHCP, TFTP, NFS, and building images is not in scope.

nfsroot mainly consists of

- scripts to configure boot payloads, chrooted on server
- scripts to make RO root image appear RW, on client
- initrd tools [RHEL6: handled by dracut]

boot payloads

PXE payload

```
/boot/pxelinux.0
/boot/pxelinux.cfg
/boot/pxelinux.msg
```

Alternative OS payload

```
/boot/freedos.img
/boot/memdisk
/boot/memtest86+-4.10
/boot/memtest86+ -> memtest86+-4.10
```

Linux payload

```
/boot/vmlinuz    -> vmlinuz-2.6.32-220.23.1.1chaos.ch5.x86_64
/boot/System.map -> System.map-2.6.32-220.23.1.1chaos.ch5.x86_64
/boot/initramfs  -> initramfs-2.6.32-220.23.1.1chaos.ch5.x86_64.img
/boot/vmlinuz-2.6.32-220.23.1.1chaos.ch5.x86_64
/boot/System.map-2.6.32-220.23.1.1chaos.ch5.x86_64
/boot/initramfs-2.6.32-220.23.1.1chaos.ch5.x86_64.img
```

Kdump payload

```
/boot/initrd-2.6.32-220.23.1.1chaos.ch5.x86_64kdump.img ->
   initramfs-2.6.32-220.23.1.1chaos.ch5.x86_64.img
```

payload reconfig

nfsroot scripts

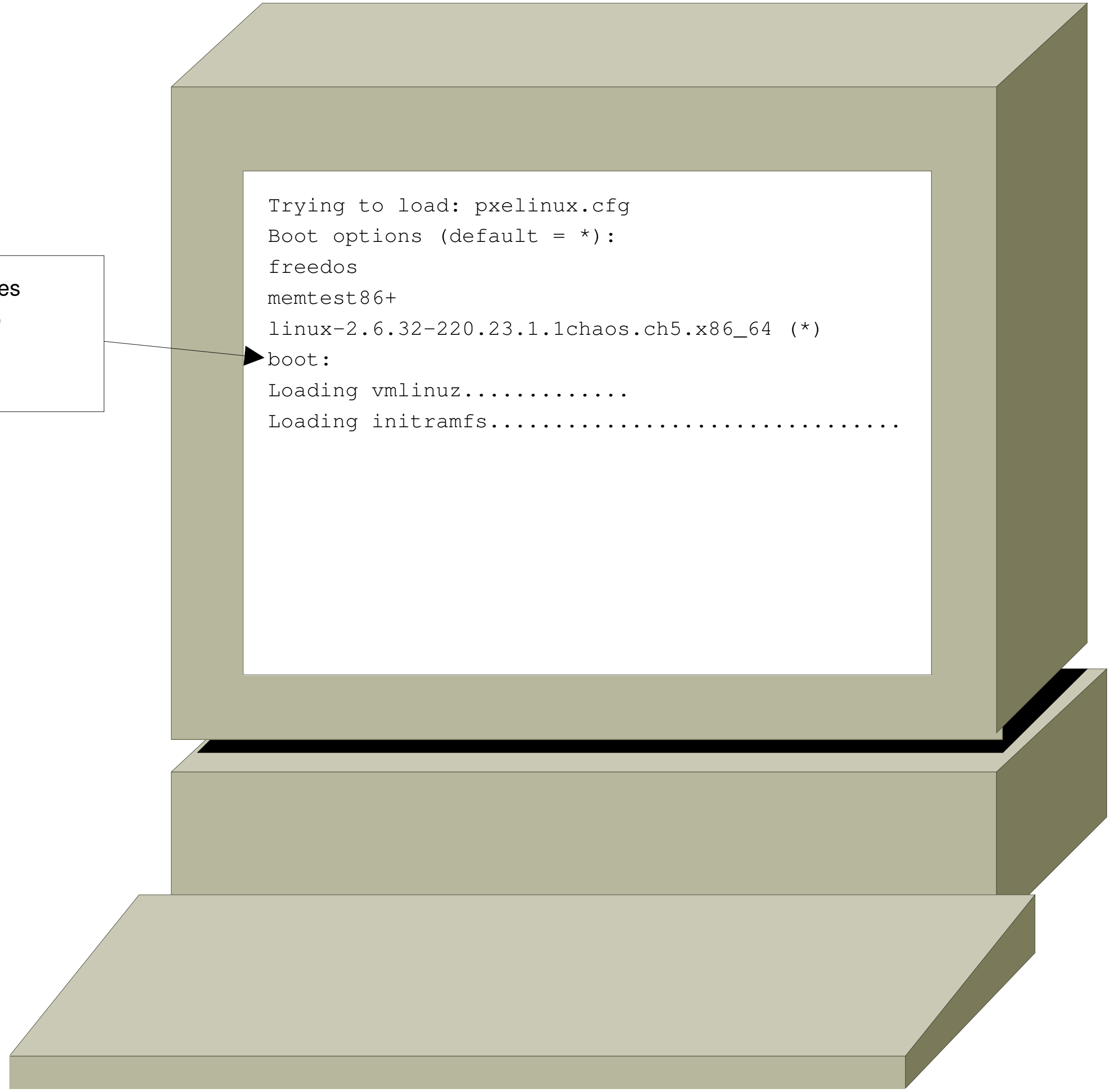
```
/usr/sbin/configpxe
/usr/sbin/nfsroot-rebuild
/usr/sbin/nfsroot-setdefault
/usr/sbin/nfsroot-kdumplinks
/usr/sbin/nfsroot-memtestlinks
```

called from grubby /sbin/new-kernel-pkg

```
/etc/kernel/postinst.d/nfsroot-postinst
/etc/kernel/prerm.d/nfsroot-prerm
```

pxelinux.0 boot prompt

pxelinux.0 pauses briefly for you to type something



```
Trying to load: pxelinux.cfg
Boot options (default = *):
freedos
memtest86+
linux-2.6.32-220.23.1.1chaos.ch5.x86_64 (*)
boot:
Loading vmlinuz.............
Loading initramfs..............................
```

making RO root appear RW

`/etc/rc.nfsroot` tries boot methods until one succeeds

- **unionfs** - overlay tmpfs
- **aufs** - overlay tmpfs
- **bind** - bind mount tmpfs copies of dirs
- **bindnfs** - bind mount NFS copies of dirs
- **rbind** - bind mount RO dirs into tmpfs /
- **ram** - copy whole root into tmpfs
- **none** - non-shared RW root
- **kdump** - save kdump to NFS and reboot
- **zram** - RO network block device root with zram overlay

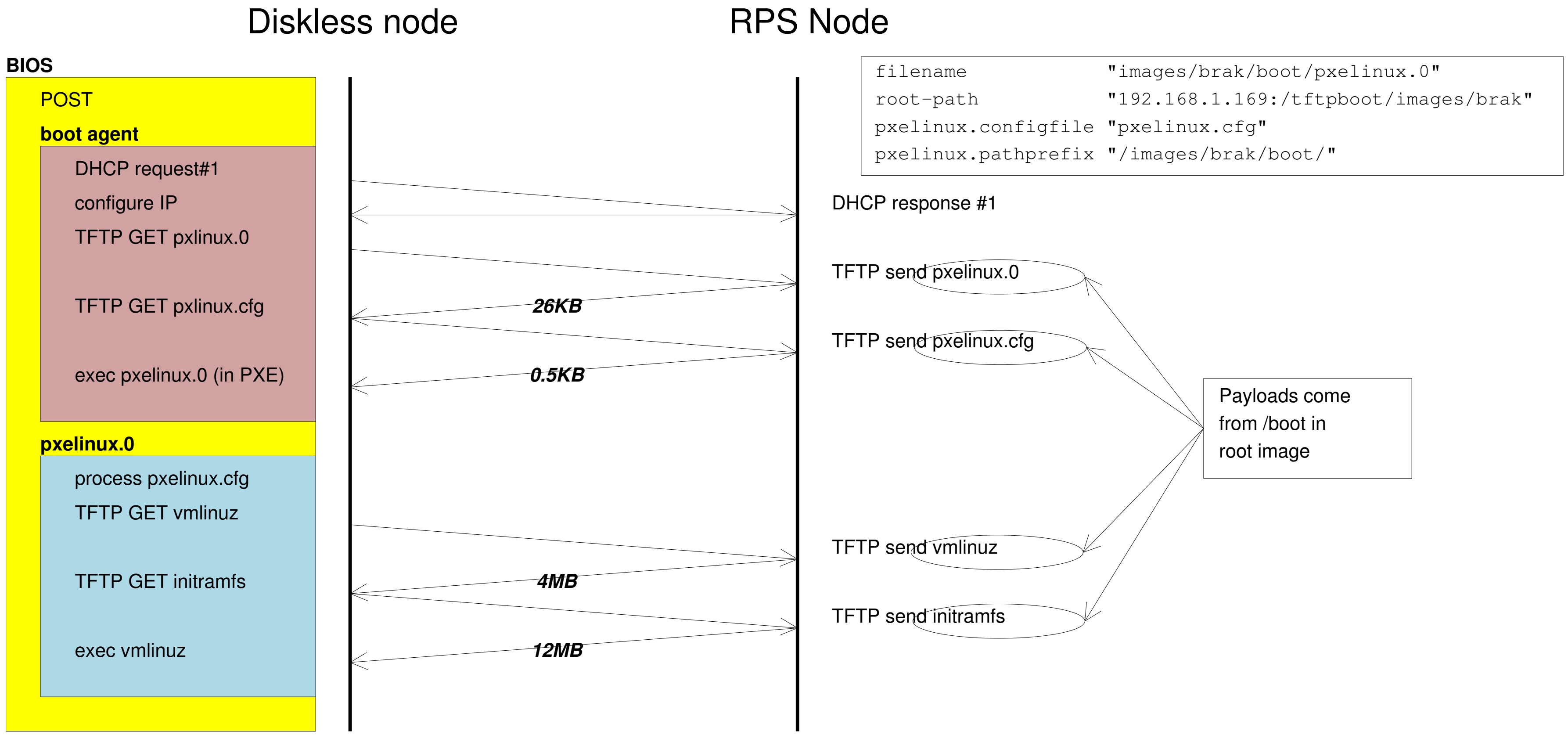
nfsroot configuration

/etc/sysconfig/nfsroot configures client boot behavior

```
# methods are tried in this order
METHODS="kdump none aufs unionfs zram bind ram"

# tmpfs max size (if tmpfs used)
#TMPFSMAX=128m

# bind only: dirs to bind mount, subdirs to not copy
RAMDIRS="/etc /var /mnt /root"
#RAMDIRS_NOCOPY="/var/cache/yum /var/lib/rpm /var/lib/yum"

# kdump config
#KDUMP_DIR=disthost:/tftpboot/dumps
#KDUMP_DIR_MOUNTOPTS="nfsvers=3,rw,nolock"
#KDUMP_LEVEL=31
KDUMP_FAILSAFE=shell
```

diskless boot sequence walkthrough

bootstrap - BIOS



Diskless node

RPS Node

```
filename            "images/brak/boot/pxelinux.0"
root-path           "192.168.1.169:/tftpboot/images/brak"
pxelinux.configfile "pxelinux.cfg"
pxelinux.pathprefix "/images/brak/boot/"
```

BIOS

POST

boot agent

DHCP request#1

configure IP

TFTP GET pxlinux.0

TFTP GET pxlinux.cfg

exec pxelinux.0 (in PXE)

pxelinux.0

process pxelinux.cfg

TFTP GET vmlinuz

TFTP GET initramfs

exec vmlinuz

DHCP response #1

TFTP send pxelinux.0

26KB

TFTP send pxelinux.cfg

0.5KB

TFTP send vmlinuz

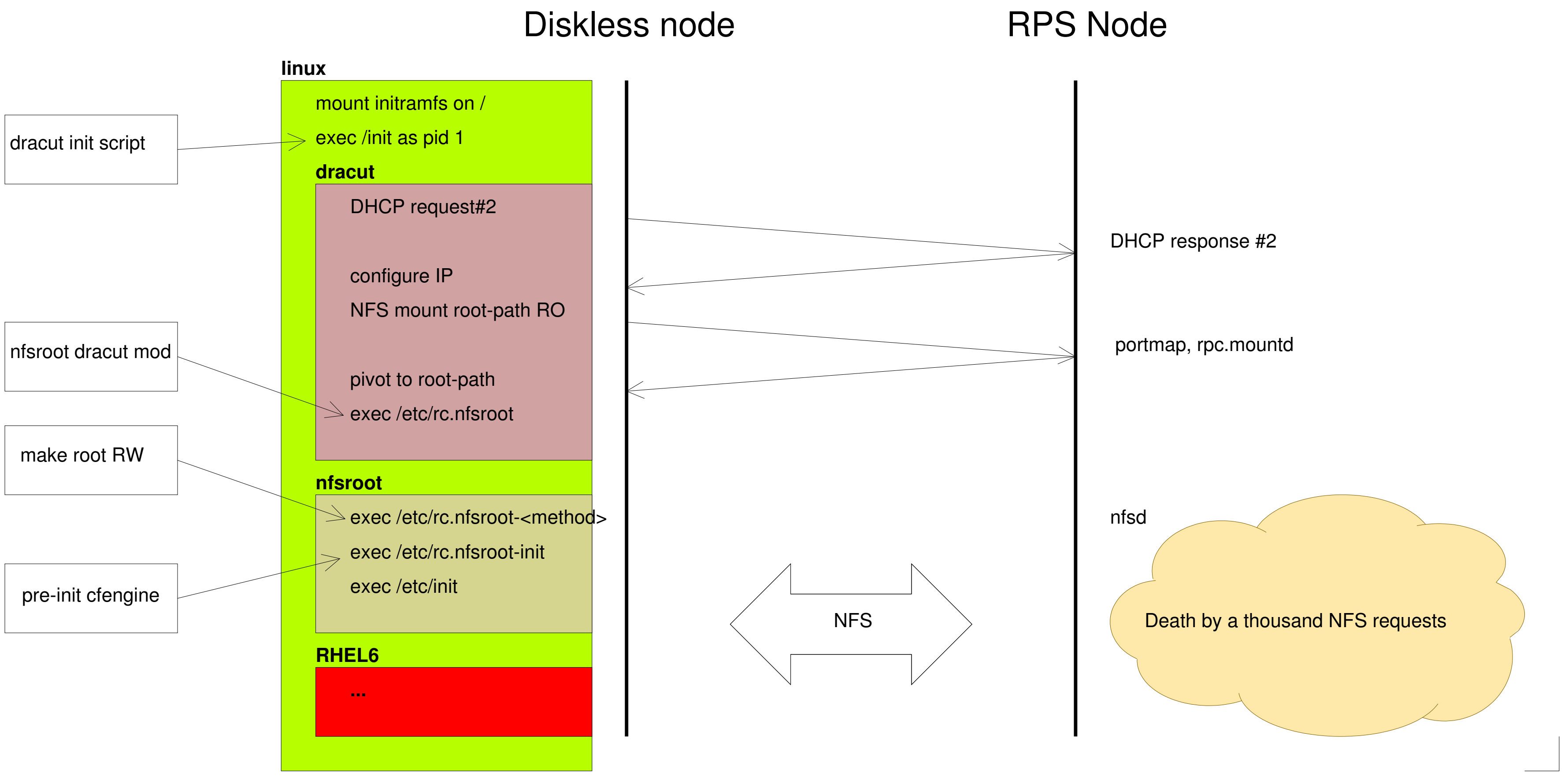
4MB

TFTP send initramfs

12MB

Payloads come from /boot in root image

bootstrap - Linux



Diskless node

RPS Node

linux

mount initramfs on /

exec /init as pid 1

dracut

DHCP request#2

configure IP

NFS mount root-path RO

pivot to root-path

exec /etc/rc.nfsroot

nfsroot

exec /etc/rc.nfsroot-<method>

exec /etc/rc.nfsroot-init

exec /etc/init

RHEL6

...

dracut init script

nfsroot dracut mod

make root RW

pre-init cfengine

DHCP response #2

portmap, rpc.mountd

nfsd

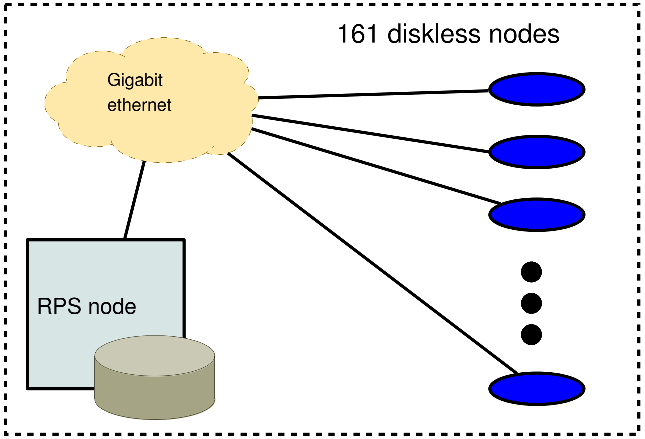
NFS

Death by a thousand NFS requests

scalability questions for large clusters

current scalability picture

TLCC2 Scalable Unit



- 5m to boot zin (18 SU * 162 = 2916 nodes)
- smaller clusters boot in about the same time

- scalable unit strategy seems to be working pretty well, but
- zin ARP table had to be hardwired
- bcasts travel widely
- RPS mirror rsync
- NFS caching not ideal

possible scalability improvements

- Multicast TFTP (RFC 2090)?

  TFTP only transfers about 16MB * 161 = 2.5GB per SU
- boot-over-Infiniband?

  Requires IB card boot agent, but b/w not really a problem.
- IPv6 Neighbor Discovery Protocol (NDP)?

  May solve ARP scalability without hardwiring.
- IP subnetting
- NFSv4 delgations?

  Reduce revalidation traffic when root image is static?
- distributed network block device?

  Enables aggressive caching and distribution.

distributed network block device

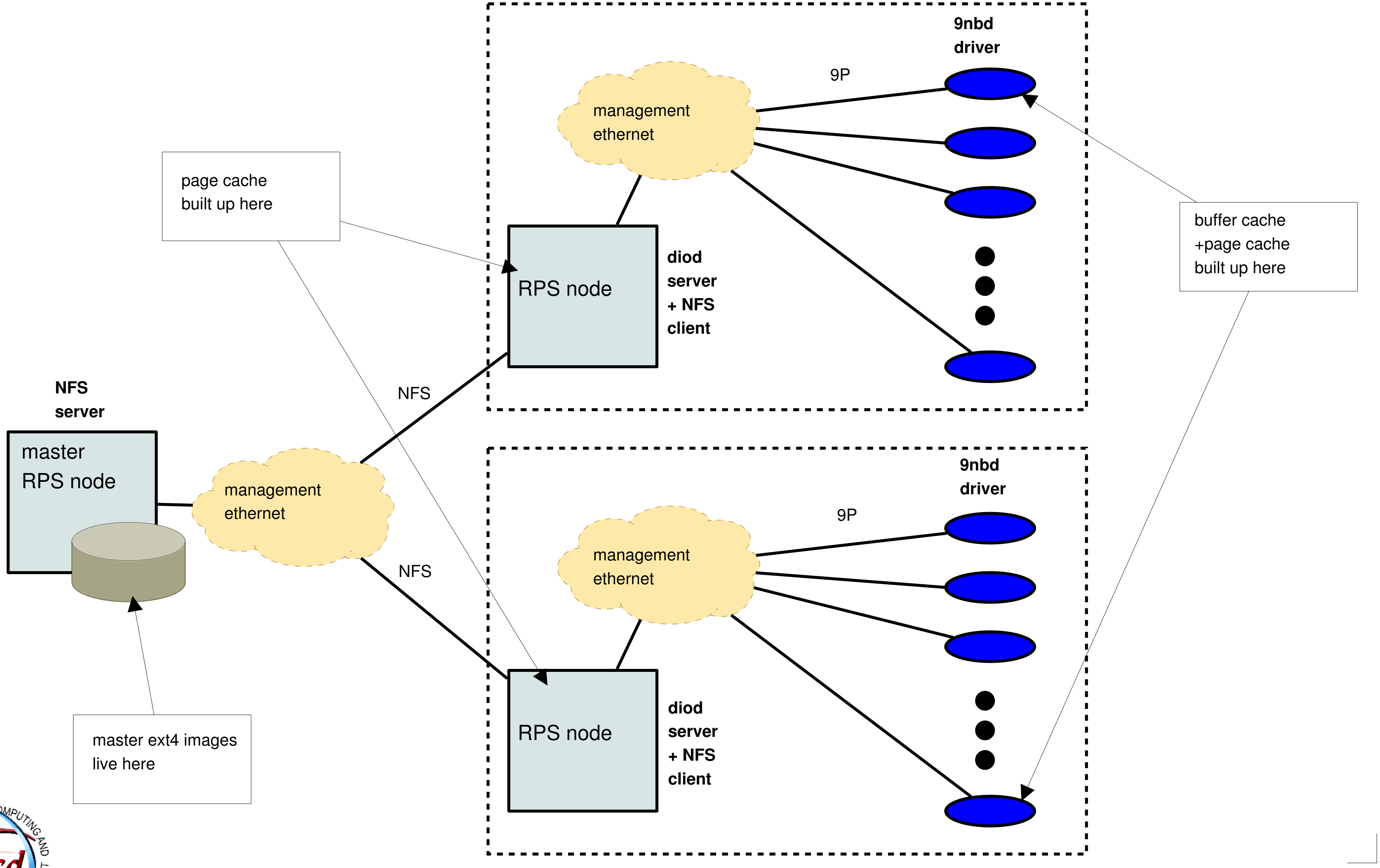
path walk: NFS scalability

- NFS does not scale well for PATH search, LD_LIBRARY search, python module search, etc
- NFS doesn't cache whole directories.
- dcache (+/-) only helps for names looked up before
- round-trip to NFS server for each unknown lookup
- NFS timeout based revalidation

path walk: block device scalability

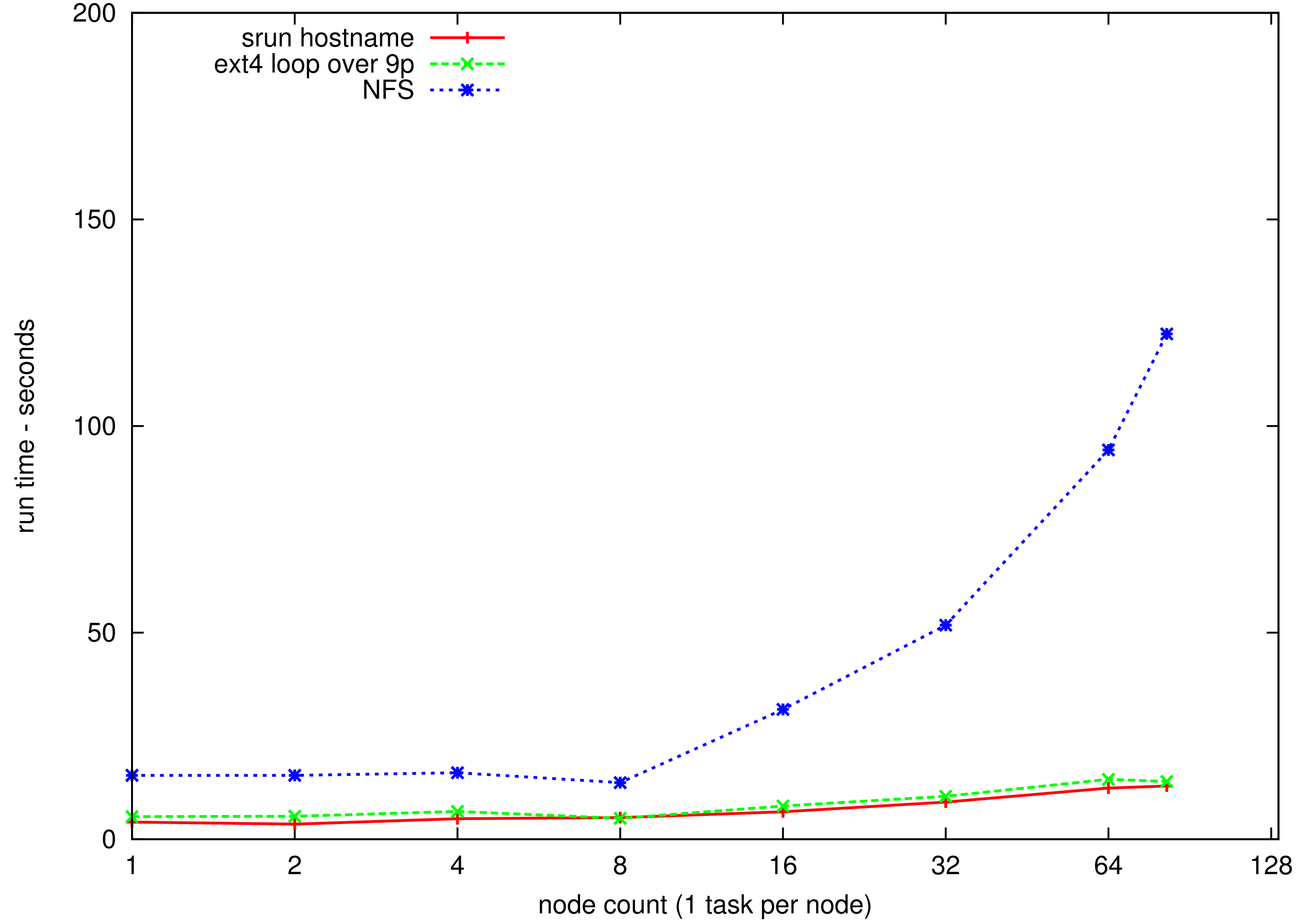
- dracut supports nbd, iscsi root
- root uses a local file system (ext4, squashfs, ...)
- nfsroot **zram** method added in 3.20
  RO block device + RW zram device using lvm-snapshot
- blocks backing directories are cached
- no revalidation
- caveat: RO root image must not change underneath diskless nodes!

distributed network block device



pathwalk test: NFS vs 9NBD

Search for 10K names in 16 dirs, each containing 10K files.


run time - seconds
node count (1 task per node)
srun hostname
ext4 loop over 9p
NFS