LIO and the TCMU Userspace Passthrough: The Best of Both Worlds

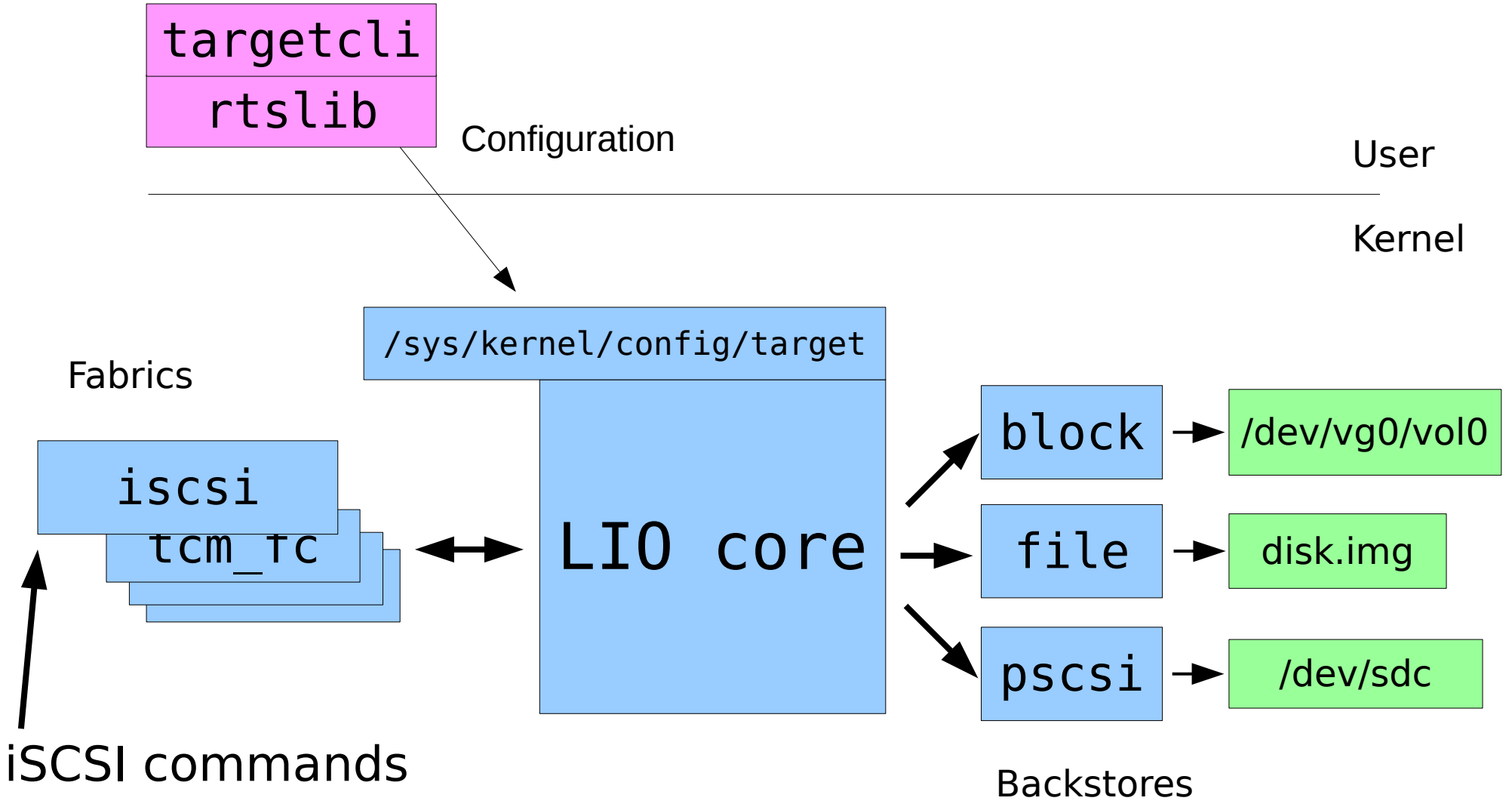
Andy Grover <agrover@redhat.com>
@iamagrover
March 11, 2015

What is LIO?

Multi-protocol in-kernel SCSI target

Multi-protocol ***in-kernel*** SCSI target

Unlike other targets like IET, tgt, and SCST, LIO is entirely kernel code.


targetcli
rtslib
Configuration
User
Kernel
/sys/kernel/config/target
Fabrics
iscsi
tcm_fc
LIO core
block
/dev/vg0/vol0
file
disk.img
pscsi
/dev/sdc
iSCSI commands
Backstores

Why add userspace command handling?

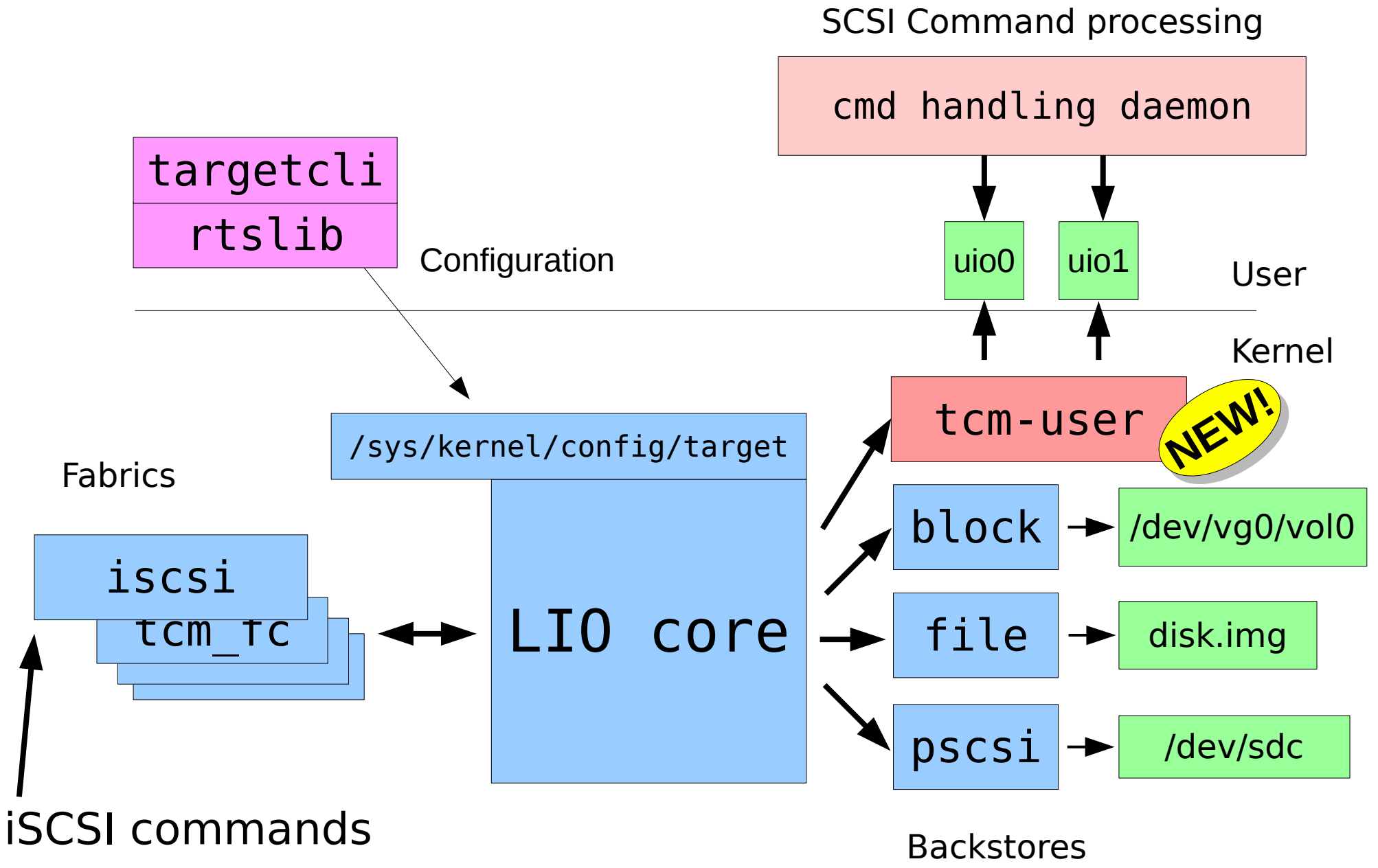
- Enable wider variety of backstores without kernel code
  - Clustered network storage filesystems like Ceph, GlusterFS, & other things that have shared libraries available
  - File formats beyond .img, such as qcow2 & vmdk for more interesting features & compatibility
  - SCSI devices beyond mass storage
  - Enable experimentation just like FUSE did for filesystems

Userspace handling challenges: perf

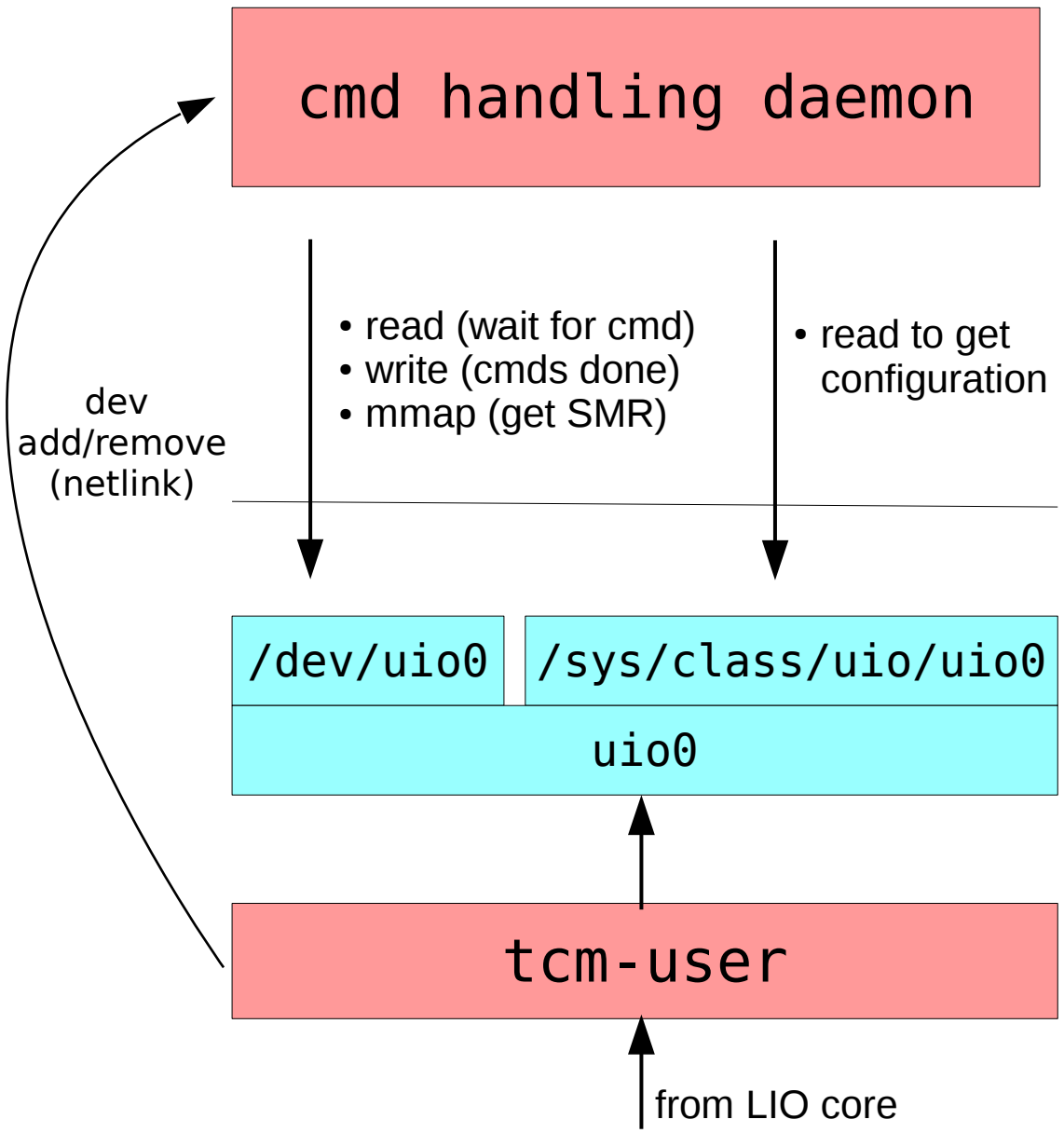
- I/O latency
- I/O throughput
- Parallelism within a dev (OoO cmd completion)
- Parallelism across devs, we're good.

Userspace handling challenges: usability

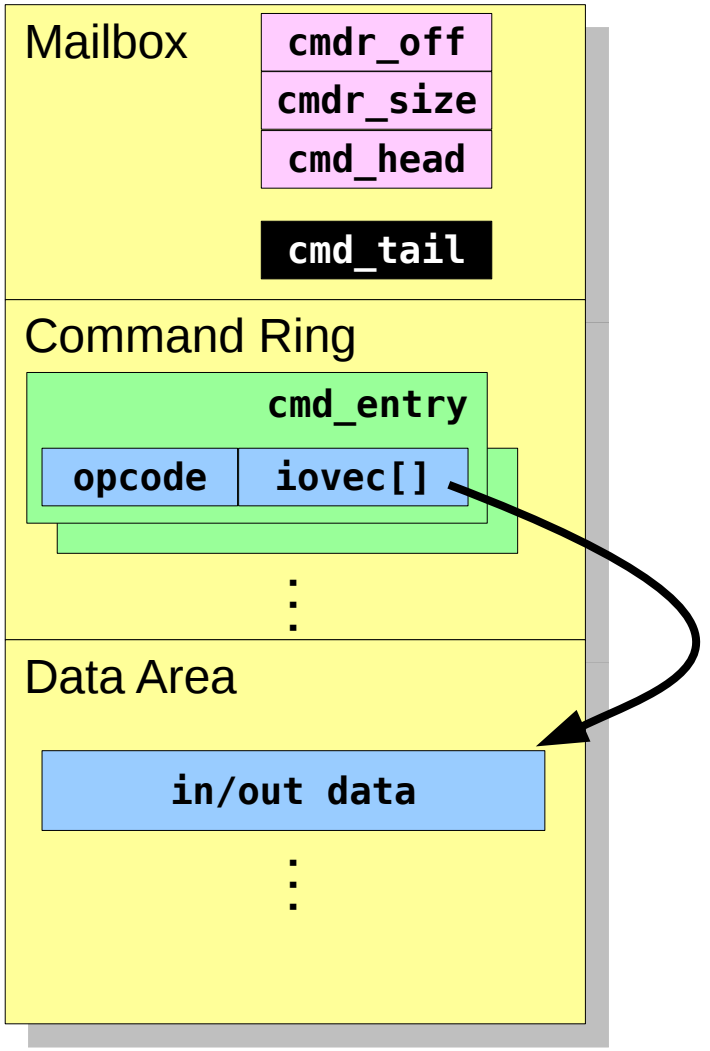
- Configuration as simple as existing backstores
- Userspace daemon failure
- Userspace daemon activation/restart
- Balance ultimate flexibility with common use
- Avoid “multiple personalities”
- Reasonable resource usage


SCSI Command processing
cmd handling daemon
targetcli
rtslib
Configuration
uio0
uio1
User
Kernel
tcm-user
NEW!
/sys/kernel/config/target
Fabrics
block
/dev/vg0/vol0
iscsi
tcm_fc
LIO core
file
disk.img
pscsi
/dev/sdc
iSCSI commands
Backstores

User/Kernel Communication



Shared Memory Region Layout
(not to scale)



tcm-user merged in 3.18

- Initial patch as simple as possible
- Performance tuning not done, BUT an interface flexible enough to not block likely perf optimization strategies
- Acceptance enabled phase 2: userspace usability pieces

Performance Opportunities for Later

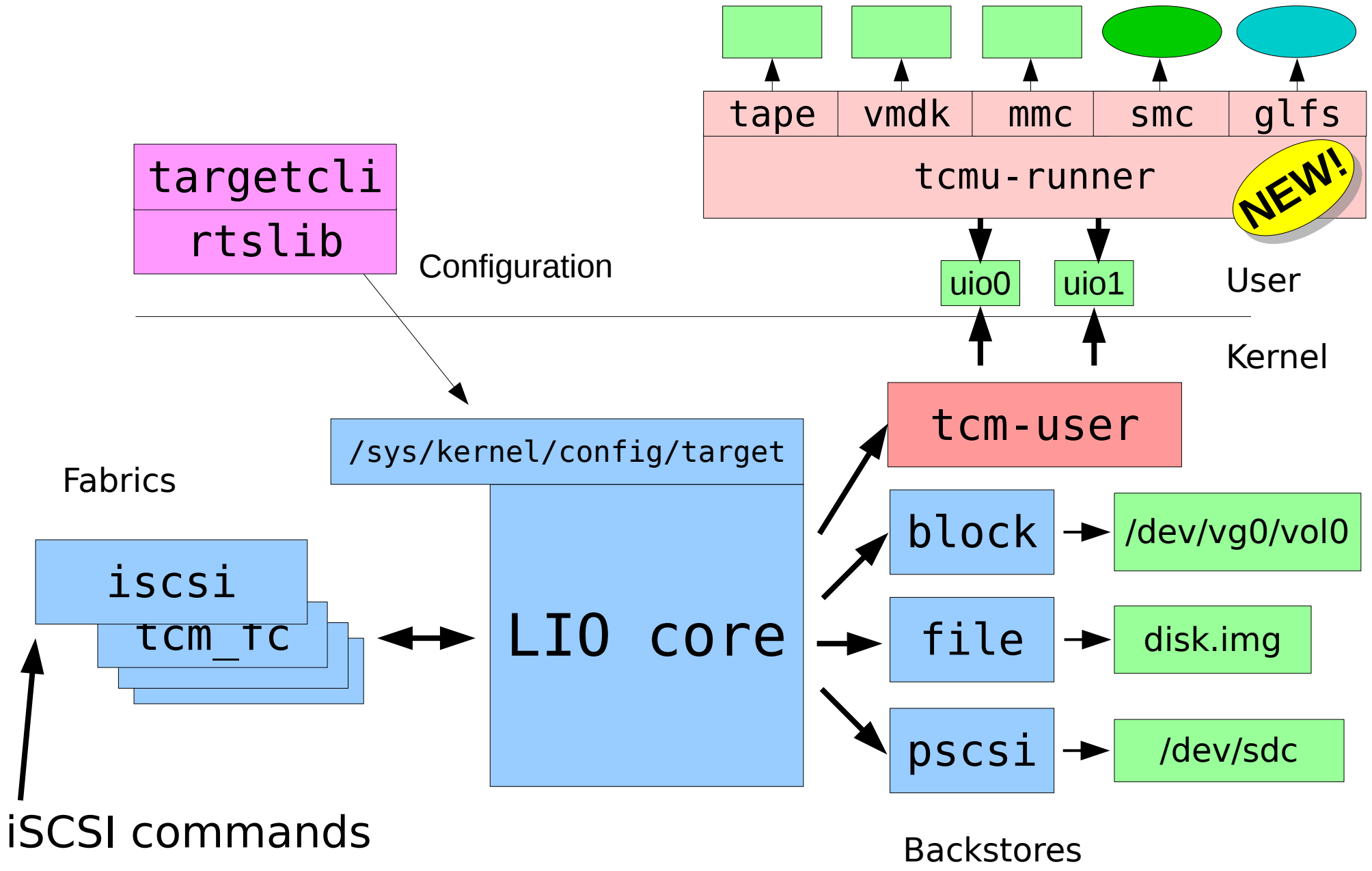
- Larger vmalloc()ed shared mem region
  - -> Demand-allocate pages to avoid bloat
    - Block size == PAGE_SIZE preferred
- Complete commands out of order
  - Must handle data area fragmentation
- Fabrics copy into already-mapped pages
  - Just moves the cache misses?
  - Fabrics don't have per-lun device queues anyway. Just IB reimplemented poorly?
- Userspace busywait on ring cmd_head

Our user-kernel API ended up flexible, but fraught with danger!

- Ring operations easy to mess up
- Make every handler write daemon boilerplate?
- And support Netlink?
- And maybe D-Bus???

tcmu-runner: A standard handler daemon

- Handle the messy ring bits
- Expose a C plugin API
- Implement library routines for common handler code, e.g. mandatory SCSI commands
- Permissively licensed: Apache 2.0
  - Doubles as sample code for do-it-yourself-ers
- Needed a prototype daemon in any case

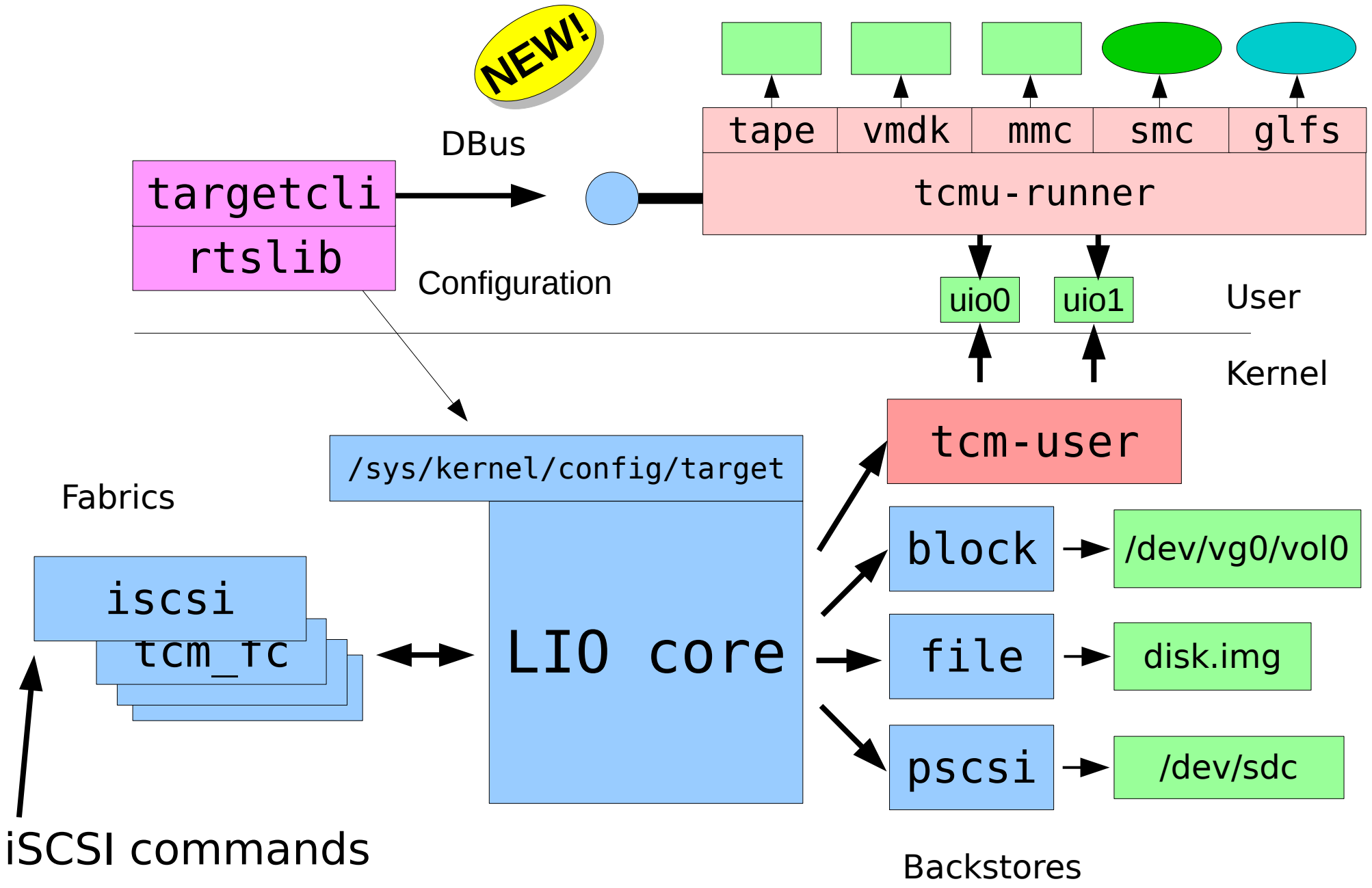

tape
vmdk
mmc
smc
glfs
tcmu-runner
NEW!
targetcli
rtslib
Configuration
uio0
uio1
User
Kernel
tcm-user
/sys/kernel/config/target
Fabrics
block
/dev/vg0/vol0
iscsi
tcm_fc
LIO core
file
disk.img
pscsi
/dev/sdc
iSCSI commands
Backstores

Handlers Need Config Info

- All configuration should still be through standard LIO mechanisms (i.e. targetcli)
- User backstore create includes a URL-like “configstring” that gives handler and per-device handler-specific stuff
- This is published in uio sysfs, and netlink add_device message
- Also has info to allow going from uio dev back to matching LIO backstore
    - Handler needs attribs, block size, etc.

Config tools need Info on Handlers Too!

- Users should call “backstores/foo create x y z” not “backstores/user <configstring>”
- targetcli needs param and help strings for xyz
- Verify backstore params are correct before creating the device
- Must be loosely coupled – no hard dependencies on targetcli *or* tcmu-handler
- Solution: D-Bus!


NEW!
DBus
tape
vmdk
mmc
smc
glfs
targetcli
rtslib
tcmu-runner
Configuration
uio0
uio1
User
Kernel
tcm-user
/sys/kernel/config/target
Fabrics
block
/dev/vg0/vol0
iscsi
tcm_fc
LIO core
file
disk.img
pscsi
/dev/sdc
iSCSI commands
Backstores

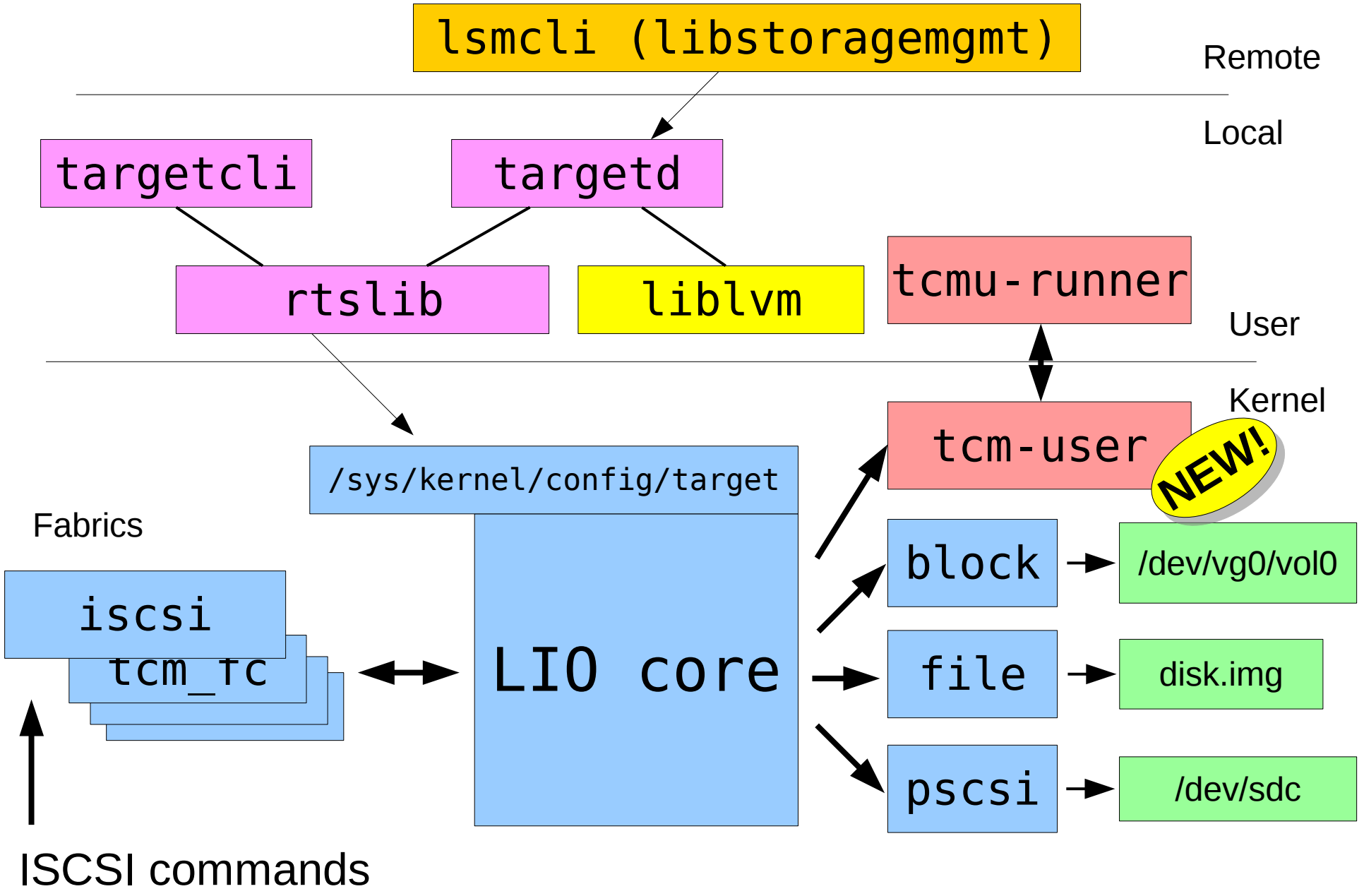
The QEMU Question

- QEMU has great support for many image formats and other backstores
- Can we reuse or integrate somehow?
- How?
  - Build qemu handler code separately and integrate as a tcmu-runner handler?
  - Extend qemu to implement TCMU directly, and possibly also enable it to configure LIO exports?
- Deferred for now

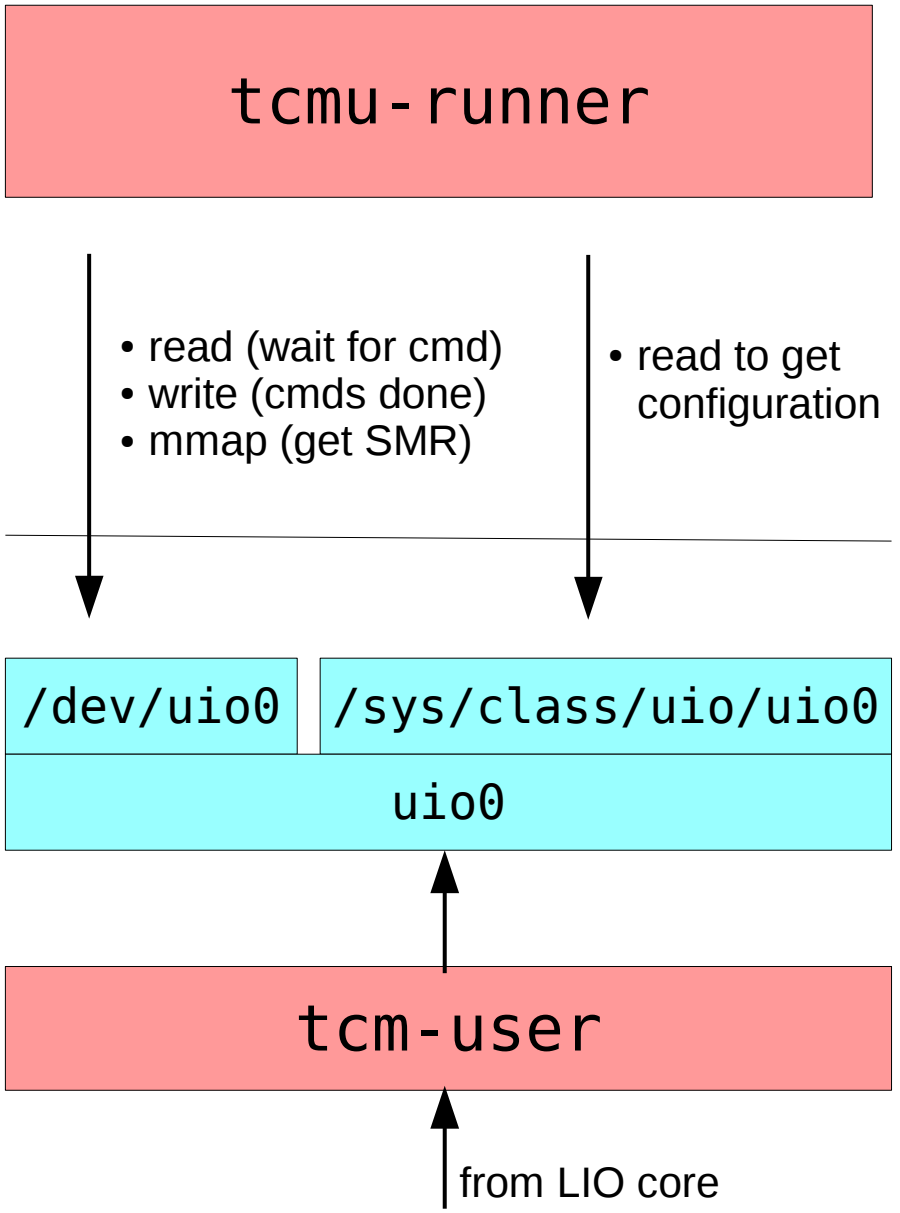
Getting Involved

- Give feedback!
  - agrover@redhat.com, target-devel@vger.kernel.org
- Check out tcmu-runner: https://github.com/agrover/tcmu-runner and its included sample handlers.
  - Use github PRs and issue tracking.
  - Much help needed, esp. QEMU hackers!
- Start doing some TCMU performance benchmarking
- Start thinking of interesting device types, userspace libraries to use, or weird things to do with a SCSI command sandbox

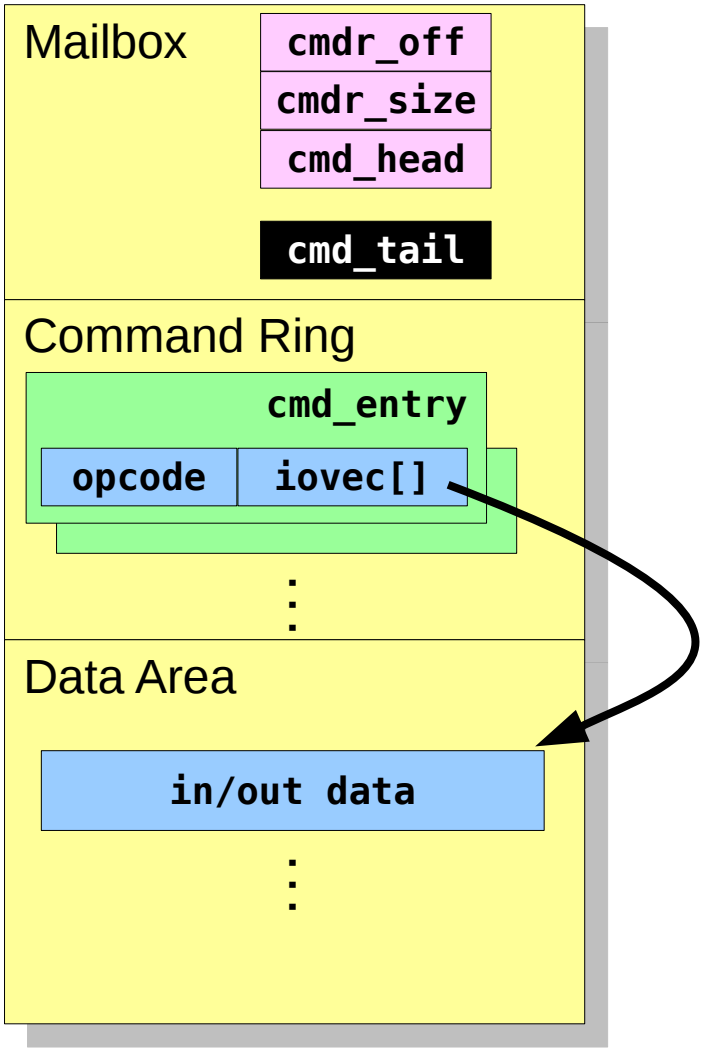
Thanks!
Questions?


lsmcli (libstoragemgmt)
Remote
Local
targetcli
targetd
rtslib
liblvm
tcmu-runner
User
Kernel
tcm-user
NEW!
/sys/kernel/config/target
Fabrics
iscsi
tcm_fc
LIO core
block
/dev/vg0/vol0
file
disk.img
pscsi
/dev/sdc
ISCSI commands

User/Kernel Communication



Shared Memory Region Layout
(not to scale)



SCSI

A set of standards for physically connecting and transferring data between computers and peripheral devices*

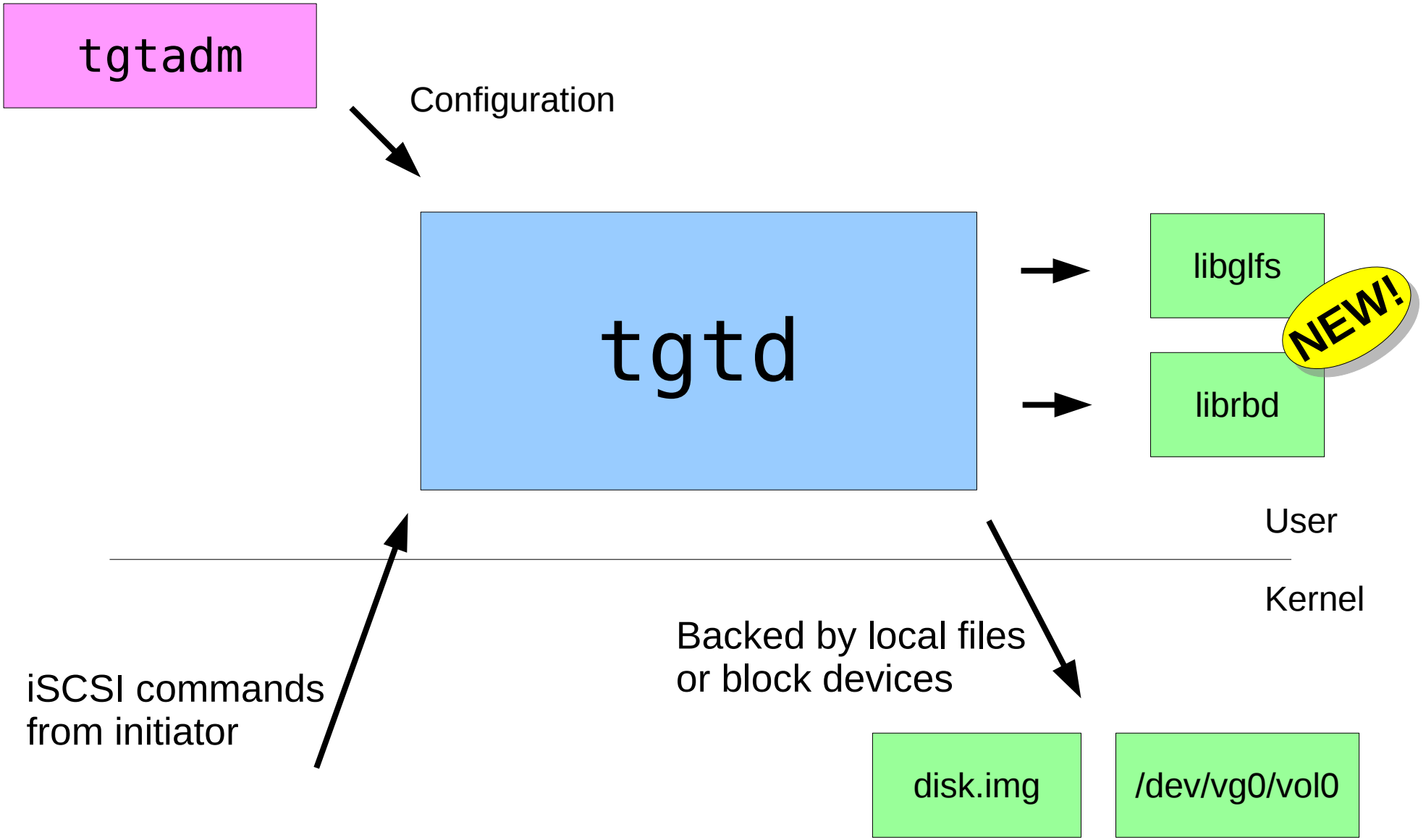
* http://en.wikipedia.org/wiki/SCSI

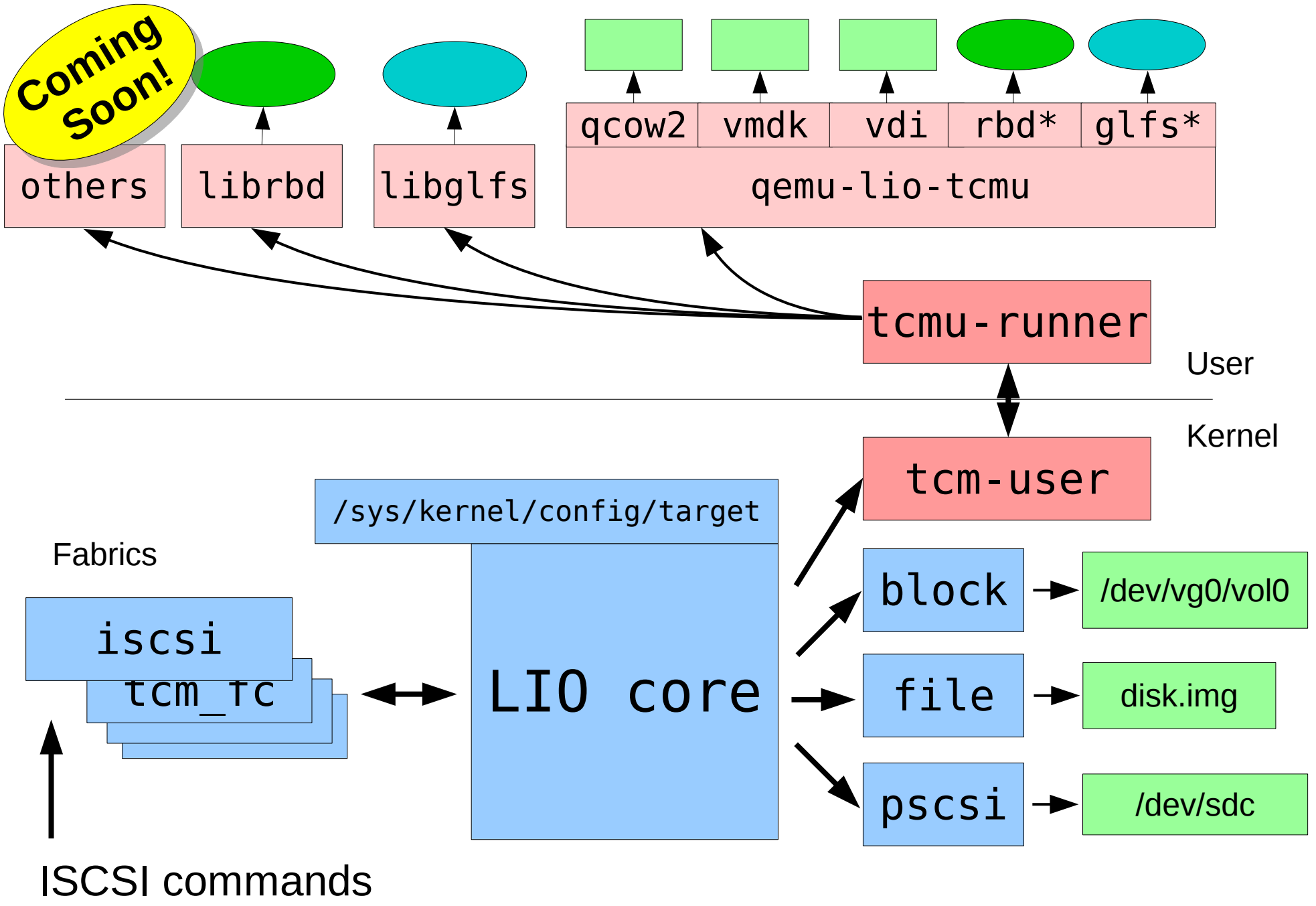
SCSI ***target***
Initiator sends commands,
Target handles them

Multi-protocol SCSI target

SCSI commands & data can be sent over many types of physical links and protocols. e.g:

- SCSI Parallel Interface (original)
- iSCSI (over TCP/IP)
- SAS (over SATA cables)
- Fibre Channel (over FCP)
- FCoE (over Ethernet)
- SRP (over Infiniband)
- SBP-2 (over Firewire)


tgtadm
Configuration
tgtd
libglfs
NEW!
librbd
User
Kernel
iSCSI commands
from initiator
Backed by local files
or block devices
disk.img
/dev/vg0/vol0


Coming Soon!
others
librbd
libglfs
qcow2
vmdk
vdi
rbd*
glfs*
qemu-lio-tcmu
tcmu-runner
User
Kernel
tcm-user
/sys/kernel/config/target
Fabrics
iscsi
tcm_fc
LIO core
block
/dev/vg0/vol0
file
disk.img
pscsi
/dev/sdc
ISCSI commands