Device-Mapper Remote Replication Target
Linux-Kongress Hamburg 2008

Heinz Mauelshagen
Consulting Development Engineer

redhat

Top

- data replication basics
- use case examples
- device-mapper architecture/overview
- device-mapper replicator architecture/overview
- mapping table syntax
- mapping table example
- dmsetup tool
- project status/future directions
- URLs

redhat

Data Replication Basics

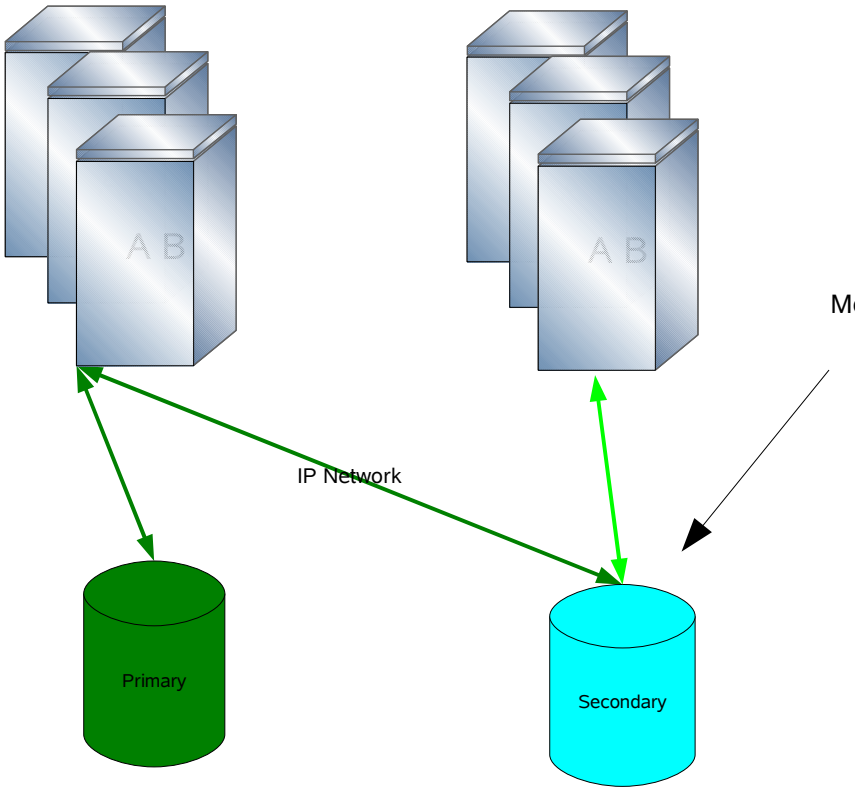
- serves disaster recovery purposes by allowing to keep copies of production data in multiple separate locations
- can occur at various possible levels (application, filesystem, block device, ...) and in active-active (read/write) access in multiple locations) and active-passive forms; this presentation focuses on block device active-passive replication
- continuous data protection (CDP) copies production data when it's being written to one or more secondary sites
- data is replicated to one or more remote sites in order to satisfy different recovery objectives
- fail-over from a primary (i.e. active) site to one of the secondary (i.e. passive) sites holding up-to-date data
- (almost) no non-redundant periods of time as with traditional, infrequent backups on a daily/weekly/... schedule
- can be synchronous (application needs to wait for replication of written data to finish) or asynchronous (application receives completion when local write completed)

redhat

Data Replication Basics (continued...)

- synchronous replication requires low-latency, high-bandwidth interconnects to minimize application write delays (e.g. 10 kilometers already cause 67μs because of the speed of light vs. 10-20μs to local cache; think about long-distance replication e.g. Berlin -> New York!)
- on long-distance links, latencies can not be made negligible and bandwidth can not be arbitrary high because of physical, technical and budget constraints
- because of that, the practical and affordable use case on long distance links is asynchronous replication
- special business requirements may cause synchronous replication on short links
- replication needs to cope with (temporary) failed remote device access
- replication properties are adjustable to address different recovery point objectives (RPOs); e.g. 100MB maximum fallbehind in asynchronous replication, hence switching to synchronous replication when that threshold's being reached
- In order to provide consistent replicas at any point in time, it is mandatory to support grouping of devices and ensure write-ordering-fidelity for the whole group
  (e.g. N database devices being written to); replication can break at any given point in time, nonetheless allowing for application recovery

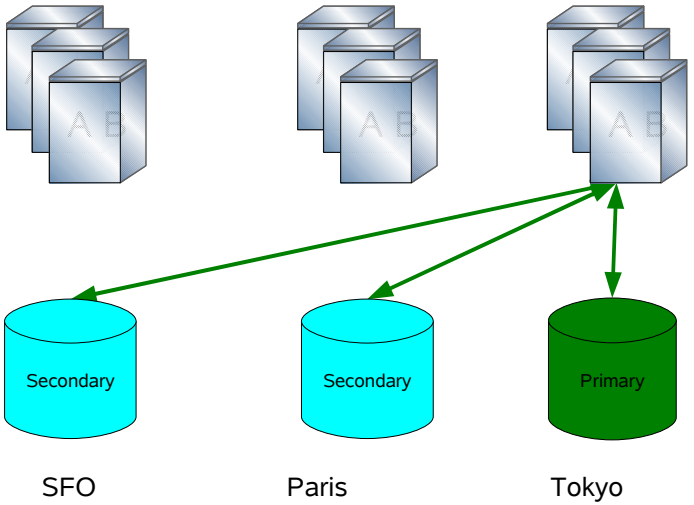
Example 1: Asynchronous Replication



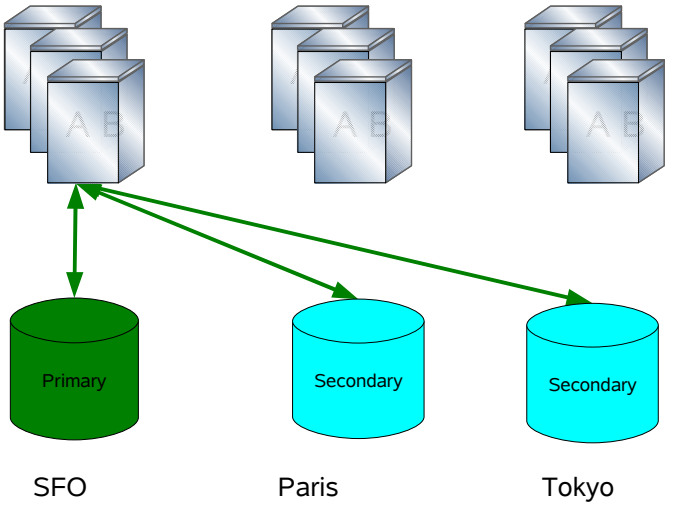
Primary owned by Single machine or Cluster

Mount only on Failure of Primary

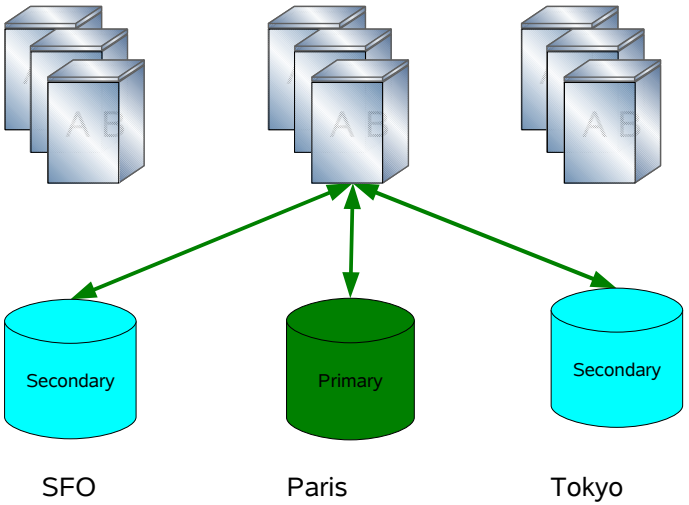
Example 2: “Follow the Sun”



00 GMT -
08 GMT

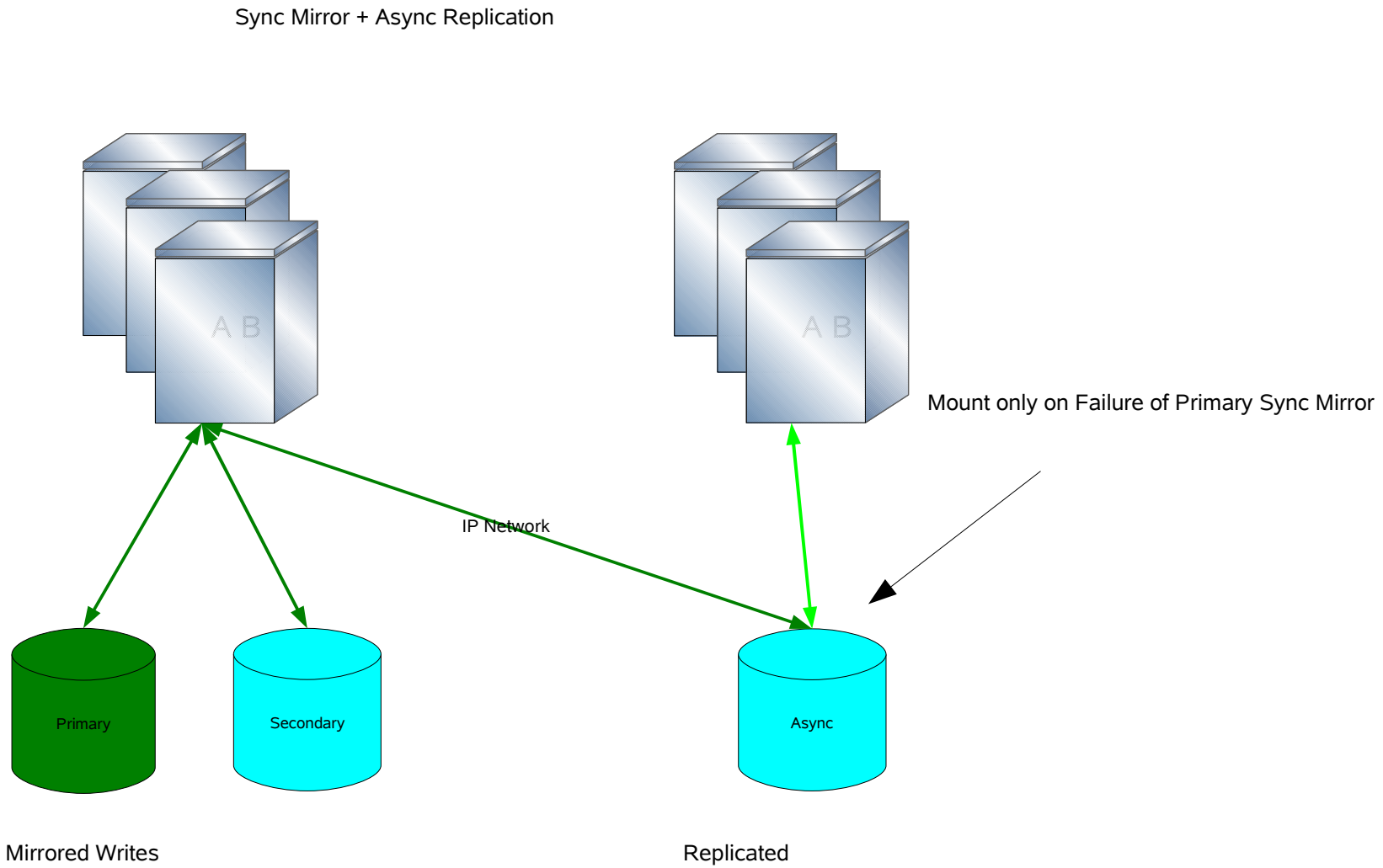


16 GMT -
24 GMT



08 GMT -
16 GMT

Example 3: Sync Mirror + Async Replication



redhat

Device-Mapper Architecture (1)

- Device-Mapper aka dm is the generic mapping runtime platform in the Linux kernel (since 2.5)
- can be used by multiple applications which require block device mapping (e.g, dmraid, LVM2, kpartx)
- doesn't “know” about any application concept of Logical Volumes etc.
- manages Mapped-Devices aka mds (create, remove, ...) with their device nodes and mappings (e.g. sector N/device A -> sector M/device B)
- defines mapping of mds via text formatted Mapping-Tables (load, unload, reload);
  format: <start> <length> <target> [<target_parameter>...]
- pluggable Mapping-Targets do the actual mapping (e.g. linear, mirror, striped, snapshot, multipath, zero, error, replicator, ...) with ctr,dtr,map,status,... interface functions
- maps to arbitrary block devices (e.g. (i)SCSI)

redhat

Device-Mapper Architecture (2)

- Mapping-Targets (e.g. dm-replicator) are dynamically loadable and register with the dm core
- mds can be stacked in order to build complex mappings (e.g. replication on top of a mirror)
- dm kernel provides dirty logging (i.e. keeping state about pending writes on regions or out-of-sync regions; dm-dirty-log), low-level io (dm-io) routines and a copy service (dm-kopyd)
- user space library (libdevmapper) to be interfaced by Device/Volume Management applications and a test tool dmsetup
- library creates device nodes to access mds in /dev/mapper/

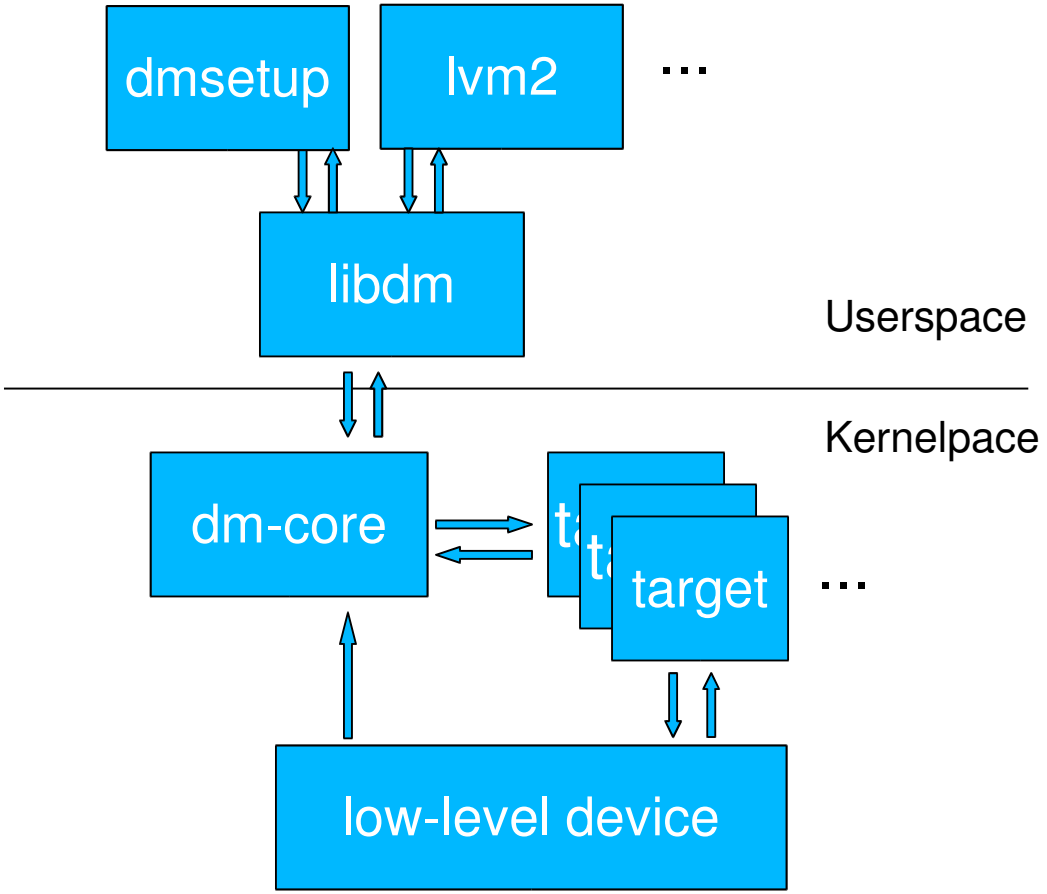
redhat

Device-Mapper Architecture (3)

- Examples of Mapping-Tables which can be activated using the dmsetup tool:
- “0 1024 linear /dev/sda1 40
  1024 2048 striped 2 64 /dev/sda1 1064 /dev/sdb1 0”
  maps sectors 0-1023 of md to /dev/sda1 at offset sector 40 using the “linear” target and
  sectors 1024-2071 onto 2 stripes with a stride size of 64 onto device /dev/sda1, sector offset 1064 and onto device /dev/sdb1, sector offset 0
- “0 1024 zero
  1024 1000 error”
  creates a “zero” mapping for sectors 0-1023 and an “error” mapping for sectors 1024-2023
- “0 83886080 mirror core 1 1024 2 /dev/sda1 64 /dev/sdb1 64”
  creates a “mirror” mapping for sectors 0-83886079 using the core dirty log with a region size of 1024 sectors and 2 mirror legs on device /dev/sda1, sector offset 64 and device /dev/sdb1, same offset

redhat

Device-Mapper Architecture Overview



redhat

dm-replicator Architecture (1)

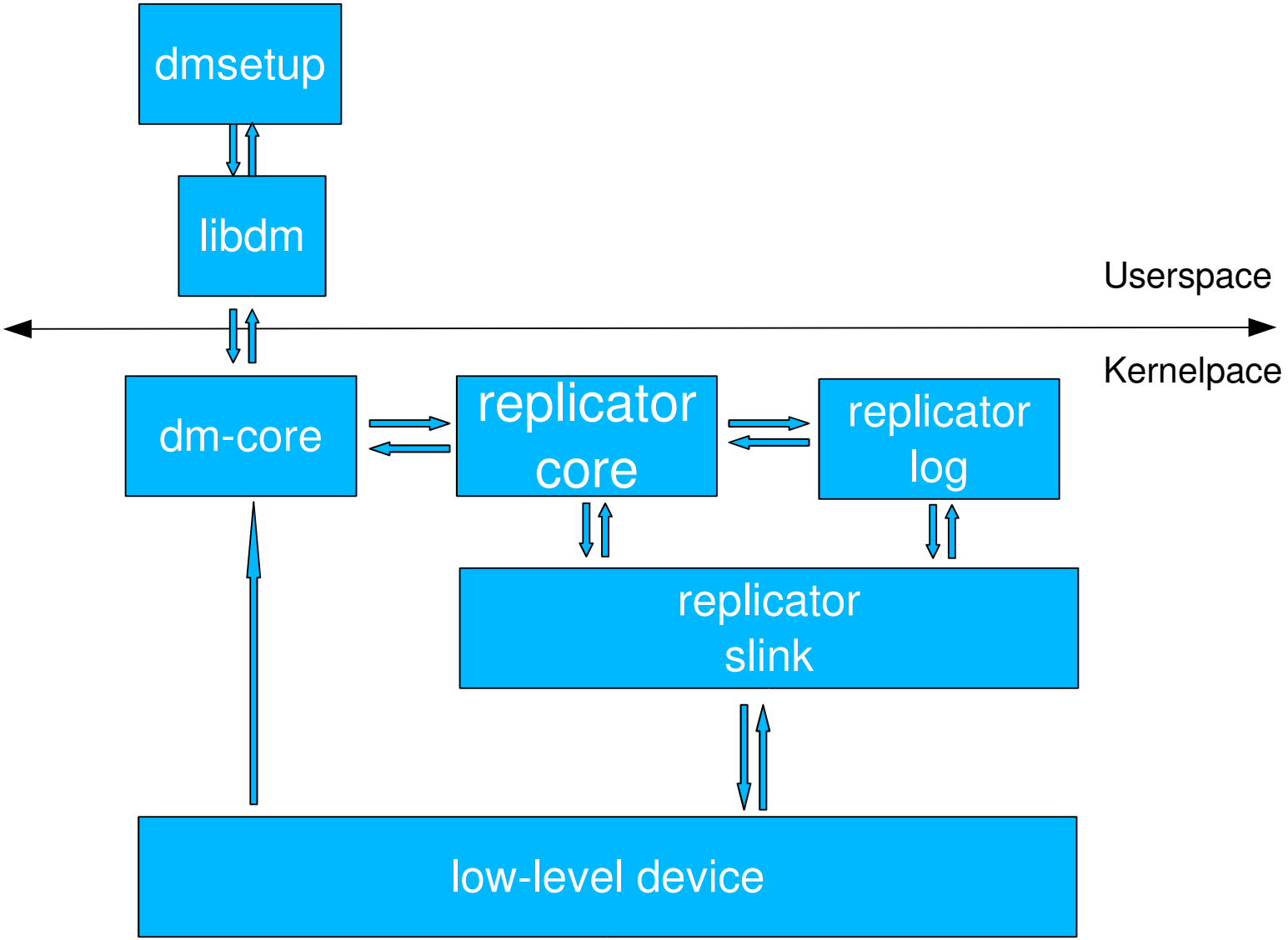
- falls appart into 3 modules: replicator core, log and slink module
- replicator core module satisfies the device-mapper core interfaces to e.g. construct/destruct a mapping, status and the actual map function to carry ios out
- a log interface abstracts storing the written data together with their replication state; different replication log plugins are possible and a “default” type with ring buffer logic has been implemented
- in case the ring buffer runs full, old entries are being dropped and state is being kept to resynchronize the data using the dm dirty log subsystem; during resynchronization there's no write-ordering guarantees
- an slink interface abstracts accessing devices on site links; different slinks plugins are possible and a “local” type has been implemented to allow for device access via device nodes; other slink modules for different transports can follow
- replicator core doesn't know how to store data in the replication log, it hands that task to the replication log plugin
- replicator log doesn't need to know how to access devices, it hands such accesses to the slink plugin

redhat

dm-replicator Architecture (2)

- sync/async and fallbehind parameters are kept by the slink module and can be requested from it
- each of the replicator modules (core, log and slink) run one or more kernel threads for various purposes
- the core has one to handle the queue of incoming ios to throttle depending on log contention
- the log has one to process incoming ios to redirect them as entries to the ring buffer together with metadata keeping state to which site links it needs copying to, hence calling the slink plugin to do the copy
- the slink has 2 to carry out copies to devices being requested by the log module and to test (temporary) failed devices in order to inform the log module when it can resubmit any failed ios
- there's additional kernel threads in the dm-kcopyd and dm-io modules being called by the slink module to do the low-level copies and ios

Device-Mapper Replicator Architecture Overview



redhat

Mapping-Table Syntax (1)

<start> <length> replicator \
replog_type #replog_params <replog_params>... \
\
slink_type #slink_params <slink_params>... \
#dev_params <dev_param>... \
dirtylog_type #dirtylog_params <dirtylog_params>...

redhat

Mapping-Table Syntax (2)

replog_type = "replog" selects the default ring buffer log
<#replog_params> = 2-4
replog_params = dev_path dev_start [auto/create/open [size]]
dev_path = device path of replication log backing store
dev_start = offset to REPLOG header
create = create a new replication log or overwrite a given one
open = open a given replication log or fail
auto = open any given replication log or create a new one
size = size to on create

redhat

Mapping-Table Syntax (3)

slink_type = "local" to handle local device nodes
#slink_params = 1-3
<slink_params> = slink# [slink_policy [fall_behind]]
slink# = used to tie the host+dev_path to a particular SLINK;
0 is used for the local link+local devices (LDs)
and 1-M are for remote links+remote devices(RDs)
slink_policy = policy to set on the slink (async/sync)
fall_behind = threshold to switch from asynchronous
to synchronous mode
(ios=N, size=N[kmgpt], timeout=N[tsmhd])

redhat

Mapping-Table Syntax (4)

#dev_params = 2
<dev_params> = dev# dev_path
dev#= unsigned int stored in the REPLOG to associate to a dev_path
dev_path = device path of device to replicate to
dirtylog_type = "-"/"core"/"disk"
#dirtylog_params = 0-3 (1-2 for core dirty log type,
3 for disk dirty log only)
dirtylog_params = [dirty_log_path] region_size [[no]sync]

Mapping-Table Example (1 LD + 1 RD, async)

```
0 209715200 replicator \
default 4 /dev/local_vg/replog_store 32 create 2097152 \
\
local 2 0 async ios=0 \
2 0 /dev/local_vg/lvol0 \
- 0
\
local 2 1 async ios=100 \
2 0 /dev/remote_vg/lvol0 \
disk 3 32768 /dev/local_vg/slink1_lvol0_dirty_log sync
```

redhat

Mapping-Table Example (2nd LD + RD, async)

```
0 209715200 replicator \
default 4 /dev/local_vg/replog_store 32 open \
\
local 2 0 async ios=0 \
2 1 /dev/local_vg/lvol1 \
- 0
\
local 2 1 async ios=100 \
2 1 /dev/remote_vg/lvol1 \
disk 3 16384 /dev/local_vg/slink1_lvol1_dirty_log sync
```

redhat

dmsetup tool

Syntax:
dmsetup create mapped_device file
dmsetup status mapped_device
dmsetup remove mapped_device
...

(say we filed the 1LD+1RD example in file “r1_def” with
 local_vg and remote_vg configured):
dmsetup create r1 r1_def
mount /dev/mapper/r1 /mnt/r1
...do_some_fs_io...
umount /mnt/r1
fsck -fy /dev/mapper/r1

redhat

Status and Future Directions

- dm-replicator in interface and code review + testing
- to go upstream ASAP (FLW)
- LVM2 design and implementation to support replication being worked on; aims to replicate groups of Logical Volumes

redhat

URLs

- http://sources.redhat.com/dm (Device-Mapper tool+library, ...)
- http://people.redhat.com/heinzm/sw/dm/dm-replicator
- http://www.redhat.com/mailman/listinfo/dm-devel
  to subscribe to dm-devel@redhat.com
- http://www.redhat.com/mailman/listinfo/lvm-devel
  to subscribe to lvm-devel@redhat.com

redhat.

Q&A