Blivet investigation report

Tao Wu

December 8, 2014

1 Introduction

Blivet is a python module for system storage configuration.

A good feature of blivet is that we can use it to model a series of changes without necessarily committing any of the changes to disk. We can schedule an arbitrarily large series of changes, seeing the effects of each as it is scheduled. Nothing is written to disk, however, until you execute the actions.

Version: 0.18.34

Installation: yum install python-blivet libselinux-python

2 What does blivet provide

Function	Support	Remark
List devices	Y	
Delete devices	Y	fail occasionally
Partition-manually	Y	
Partition-automation	Y	Not verified
Format	Y	
Mount	Y	
Unmount	Y	fail occasionally
Discover ZFCP disks	Y	
Discover SCSI disks	?	
Create PV	Y	
Create VG	Y	
Create LV	Y	
Resize	Y	Not verified
Encrypt	Y	Not verified

Arch supported:

Cell	Y
Mactel	Y
Efi	Y
X86	Y
PPC	Y
S390	Y
IA64	Y
Alpha	Y
Sparc	Y
AARCH64	Y
ARM	Y

Filesystem supported:

Filesystem	Support
Ext2FS	Y
Ext3FS	Y
Ext4FS	Y
FATFS	Y
EFIFS	Y
BTRFS	Y
GFS2	Y
JFS	Y
ReiserFS	Y
XFS	Y
HFS	Y
AppleBootstrapFS	Y
HFSPlus	Y
NTFS	Y
NFS	Y
NFSv4	Y
Iso9660FS	Y
NoDevFS	Y
DevPtsFS	Y
ProcFS	Y
SysFS	Y
TmpFS	Y
BindFS	Y
SELinuxFS	Y
USBFS	Y

3 Basic methods we may need

Blivet supplies a great many methods to manipulate storage devices recognized by kernel. In the first instance, I think we should focus on the following ones which are sufficient enough to implement the functions refereed in section two.

- **blivet.Blivet.reset()**
  -Reset storage configuration to reflect actual system state.

- **blivet.devicetree.DeviceTree.getDeviceByName(name, incomplete=False, hidden=False)**
  -Return a device with a matching name.
  Parameters:
  name (str) - the name to look for
  incomplete (bool) - include incomplete devices in search
  hidden (bool) - include hidden devices in search
  Returns: the first matching device found
  Return type: device

- **blivet.Blivet.recursiveRemove(device)**
  -Remove a device after removing its dependent devices.
  We use this method to destroy devices which are not leaves.

- **blivet.Blivet.initializeDisk(disk)**
  -(Re)initialize a disk by creating a disklabel on it.
  Parameters:
  disk (StorageDevice) - the disk to initialize
  Returns: None
  Raises: ValueError

- **blivet.Blivet.newPartition(*args, **kwargs)**
  -Return a new (unallocated) PartitionDevice instance.
  Parameters:
  fmt_type (str) - format type
  fmt_args (dict) - arguments for format constructor
  mountpoint (str) - mountpoint for format (filesystem)

- **blivet.Blivet.createDevice(device)**
  -Schedule creation of a device.
  Parameters:
  device (StorageDevice) - the device to schedule creation of
  Return type: None

- **blivet.formats.getFormat(fmt_type, *args, **kwargs)**
  - Return an instance of the appropriate DeviceFormat class.
  Parameters:
  fmt_type (str) - The name of the formatting type
  Returns: the format instance
  Return type: DeviceFormat
  Raises: ValueError

- **blivet.partitioning.doPartitioning(storage)**
  -Allocate and grow partitions.
  Parameters:
  storage (Blivet) - Blivet instance
  Raises: PartitioningError
  Returns: None

- **blivet.Blivet.formatDevice(device, fmt)**
  -Schedule formatting of a device.
  Parameters:
  device (StorageDevice) - the device to create the formatting on
  fmt - the format to create on the device
  Return type: None
  A format destroy action will be scheduled first, so it is not necessary to create and schedule an ActionDestroyFormat prior to calling this method.

- **blivet.Blivet.newVG(*args, **kwargs)**
  - Return a new LVMVolumeGroupDevice instance.
  Parameters:
  name (str) - the device name (generally a device nodes basename)
  exists (bool) - does this device exist
  parents (list of StorageDevice) a list of parent devices
  sysfsPath (str) - sysfs device path
  peSize (Size) - physical extent size
  Returns: the new volume group device
  Return type: LVMVolumeGroupDevice

- **blivet.Blivet.newLV(*args, **kwargs)**
  - Return a new LVMLogicalVolumeDevice instance.
  Parameters:
  fmt_type (str) - format type
  fmt_args (dict) - arguments for format constructor
  mountpoint (str) - mountpoint for format (filesystem)
  thin_pool (bool) - whether to create a thin pool
  thin_volume (bool) - whether to create a thin volume

Returns: the new device
Return type: LVMLogicalVolumeDevice

- **blivet.Blivet.doIt()**
  -Commit queued changes to disk.
  -Caution: Data on devices may be destroyed after this method succeeds.

- **blivet.zfcp.ZFCP.addFCP(devnum, wwpn, fcplun)**
  -Manully online ZFCP/SCSI disks.
  Parameters:
  devnum (str) - scsi disk number
  wwpn (str) - world wide port name
  fcplun (str) - logical unit number
  Return type: None

- **blivet.devicetree.populate(cleanupOnly=False)**
  - Locate all storage devices.
  Loop and scan for devices to build the device tree.

4 A simple example on LPAR11

```
import blivet

b=blivet.Blivet()
b.reset()

#initialize disk
dasdb=b.devicetree.getDeviceByName("dasdb")
b.recursiveRemove(dasdb)
b.initializeDisk(dasdb)

#create new PV
part=b.newPartition(size=8000,parents=[dasdb])
b.createDevice(part)
fmt=blivet.formats.getFormat("lvmpv",lable="pv1")
b.formatDevice(part,fmt)
blivet.partitioning.doPartitioning(b)
b.doIt()

# create new VG
a=b.pvs
request=b.newVG(parents=a)
```

```
b.createDevice(request)
b.doIt()

#create new LV
a=b.vgs
request=b.newLV(size=2500,name="lv1",parents=a)
b.createDevice(request)
fmt=blivet.formats.getFormat("ext4",lable="test1",
     mountpoint="/home/test1")
b.formatDevice(request,fmt)
blivet.partitioning.doPartitioning(b)
b.doIt()

fmt.mount()

#discover scsi devices
a=b.zfcp
a.addFCP("0.0.5080","0x500507630300c52a",
                    "0x4013401300000000")
b.devicetree.populate()
```

5 Discovery of ZFCP disks

To manually online ZFCP disks, first of all we invoke the addFCP() methods refered in Section three. We can also put these arguments in a configure file as anaconda does. Then we use populate() to add the ZFCP disk to devicetree so that blivet can see it.

In order to get the arguments needed, we can use some tools such as:
udevadm info -a -p /sys/class/scsi_generic/sg0

6 What to do in the next stage

1. Do as much tests as more to figure out what cause some functions refered in section two fail, for example, what is the device status on the spot (encrypted? mounted? have some children devices? have some special data on it being used? etc.)
2. Continue to explore blivet codes and dig out some useful methods.