Stratis: A New Approach to Local Storage Management

March 22, 2017
Andy Grover <agrover@redhat.com>
https://stratis-storage.github.io/
https://github.com/stratis-storage
Volume Management Choices on Linux Today?
Characteristics of Volume-managing Filesystems (VMFs)

- Multiple logical fs trees from a single shared storage resource
  - Size not specified, only what you need
  - Copy-on-write, snapshots
- Spanning of multiple block devices
- Integrated UI
- Integrated implementation of features
  - RAID, snapshots, caching, compression, deduplication
Stratis Goals

• Deliver VMF features, but build on existing kernel capabilities

• Go beyond VMF features
  1) Further minimize concepts and complexity for the user
  2) Maximize flexibility
  3) Programmatic API
  4) Active management and monitoring
1) Minimize Concepts and Complexity

```
Block device
Block device
Block device
```

```
Pool
```

```
Filesystem
Filesystem
Filesystem
Filesystem
```
2) Maximize Flexibility

- Efficient use of differing-capacity drives
- Support adding single drives to a redundant pool
- Re-establish redundancy after any drive is removed
- 1 to 1000 drives in a pool
- 1M+ filesystems
Flexibility vs. Performance: SSDs Change Everything

• Adding SSDs is usually much better than optimizing HDDs for performance
  – Stratis supports a cache tier
  – Primary (data) tier focuses on integrity, redundancy, flexibility, and ease of management over performance
3) D-Bus API

Direct programmatic access using a language-independent interface

- Anaconda
- stratis-cli
- Cockpit
- Docker
- libvirt
- Gluster
- Ceph

Vault 2017
github.com/stratis-storage
4) Management and Monitoring

- Internal monitoring
  - Self-monitoring, scrubbing, rebalancing
- External alerts
  - Event driven, via API
  - Notifications sent where someone sees them
Stratis In Detail

- **stratis-cli**
- **stratisd**
  - Manages & Monitors internal components
- **filesystem**
  - xfs
  - dm-thin
- **Pool**
  - dm-thinpool
    - meta dev
    - data dev
  - dm-cache
    - SSD
    - SSD
  - dm-raid
    - dm-integ.
    - dm-integ.
    - dm-integ.
  - none

Extends & trims fs as needed

Vault 2017
github.com/stratis-storage
## Required and Optional Layers, Current and Future Plans

<table>
<thead>
<tr>
<th>Current, Layering Defined</th>
<th>Current, Layering TBD</th>
<th>Future, Layering TBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>XFS</td>
<td>dm-crypt</td>
<td>dm-multipath</td>
</tr>
<tr>
<td>dm-thin</td>
<td>dm-cache</td>
<td></td>
</tr>
<tr>
<td>dm-thinpool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dm-raid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dm-integrity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block Devices</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Vault 2017
github.com/stratis-storage
The Flex Layer

XFS
thin
thin-pool
metadata-dev
data-dev
linear
linear
linear
block device
block device

fs containing upper-layer data
DEMO
Stratis: Summary

- Agglomerates block devices into pools
- Creates filesystems from the pools
- Encapsulates other advanced features to make features easy to use
- Builds on existing Linux capabilities
- Daemon, D-Bus API, and command-line tool
Current Project Status

• Draft design doc & API reference
• Development underway  
• Still early! Seeking collaborators for design, development, review, docs, testing, & feedback from potential users
Thanks!

Documentation: https://stratis-storage.github.io/

Code and Issue Tracking: https://github.com/stratis-storage

IRC: freenode #stratis-storage

Mailing list: stratis-devel@lists.fedorahosted.org

https://lists.fedoraproject.org/admin/lists/stratis-devel.lists.fedorahosted.org/
Attic
Rust, why?

- Development productivity
  - Static typing → easy refactoring
  - No time spent chasing segfaults
  - Great pkg and build system
  - High-level programming language
    - Containers, iterators, closures, sum types

- Compiled, no lang runtime
  - Needed for early boot environment
Create multiple RAID devices across the member drives

- 1TiB
- 2TiB
- 2TiB
- 3TiB

`dm-raid #1`

`dm-raid #2`

Unused
Stratis Abstraction “Leaks”

- Stratis-managed Filesystems will have a size and be recognizably of a fs type such as XFS
  - User should not modify directly!

- ‘stratisd’
  - User should not kill!
HW trends towards greater self-management

- MFM, RLL → IDE: Hide physical on-disk data encoding and drive mechanism
- Specify size/CHS of drive in BIOS (succ IDENTIFY)
- Parking your heads (autopark)
- Master/slave jumpers, or SCSI termination
- Bad block remapping
SW trends towards fewer up-front decisions, less maintenance

- Volume size flexibility
  - Step 1: LVM. More easily enlarge volumes (and then online growfs)
  - Step 2: Thin provisioning
    - Grow or reclaim space from a fs
    - FS possibly unaware

- Fragmentation-resistant (less maintenance)

- RAID (redundancy → more reliability)
Btrfs Blockdev flexibility

• RAID
• RAID reshape: adding/removing devices
  – “rebalance” command
• Max disks = ?
• Raid1 supports different-sized drives
  – Raid56 = ?
• Raid1 on a single disk