WalB: A Fast and Low Latency Backup System for Block Devices

Open Source Summit Japan 2017
Kota Uchida
June 1, 2017
About me

- Kota Uchida
- SRE team at Cybozu, Inc.
- A WalB developer
About Cybozu

- A large cloud service vendor in Japan.
- Largest market shares in field of collaborative software.

- We serve web applications on our own cloud platform.
  - kintone: a low-code business app platform
  - and more
#customer companies: 19,000+

#accesses / day: 190 millions

write IOs / day: 24.5 TiB
Service Level Objective

- **24/7** nonstop service
- **99.99%** availability (4 min / month)
- Daily backup (retention period is **14 days**)
- Disaster recover: copy data to a remote site **once a day**
The scope of this talk
Snapshot Management with dm-snap

Logical Structure

<table>
<thead>
<tr>
<th>Logical Structure</th>
<th>Physical Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snapshot Image</td>
<td>Original Volume Area</td>
</tr>
<tr>
<td></td>
<td>Snapshot Area</td>
</tr>
<tr>
<td>A</td>
<td>A'</td>
</tr>
<tr>
<td>B</td>
<td>B'</td>
</tr>
</tbody>
</table>

Write A'          Write B'

Latest Image

A' B'

Physical Structure

(1) CoW

(2) Write

Original Volume Area

A' B'

Snapshot Area

Mapping Info

A B
Backup using dm-snap

Logical Structure

Snapshot0

(1) Full-scan an old snapshot

A  B

Snapshot1

(2) Full-scan a new snapshot

A'  B'

(3) Generate a diff image by comparing two snapshots

A'  B'
Full-scan at night

- Read throughput of storage devices
- Backup processing time
- User request rate
- Daytime
UX degradation during a full-scan

90 percentile of request response time [milliseconds]
We have no more “nights”

Until now:
Full scan is allowed only when access rate is low, i.e., at night.

From now on:
We have to handle accesses from multiple timezones.

We must be able to backup any time without UX degradation.
New Solution

- We need a new solution with:
  - No IO spikes
  - Short backup time

- We compared dm-thin with WalB
What is dm-thin?

- dm-thin provides thin-provisioning volume management to
  - share same data among volumes
  - reduce disk usage using snapshots
- In the mainline Linux kernel
Snapshot Management with dm-thin

Logical Structure

Latest Image

Physical Structure

Latest Tree

A
Snapshot Management with dm-thin

**Logical Structure**

<table>
<thead>
<tr>
<th>Snapshot</th>
<th>Latest Image</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
</tbody>
</table>

**Physical Structure**

- **Snapshot Tree**
- **Latest Tree**

![Diagram showing the logical and physical structure of snapshot management with dm-thin.](Image)
Snapshot Management with dm-thin

Logical Structure
- Snapshot
  - A
- Latest Image
  - A'

Physical Structure
- Snapshot Tree
- Latest Tree
  - (1) CoW
  - (2) Update

A

(1) CoW

A'

(2) Write
Backup using dm-thin

Logical Structure
- Snapshot0: A, B
- Snapshot1: A', B'

Physical Structure
- Snapshot0
- Snapshot1

Generate a diff image using dm-thin metadata
What is WalB?

- A real-time and incremental backup system
  - developed at Cybozu Labs
- Can backup block devices without IO spikes
Special Block Devices for WalB

Any application (File system, DBMS, etc.)

WalB device

Read

Data device

Linear mapped

Write

Log device

Ring buffer
Write IO Logging and Backup with WalB

Time series of write I/Os

Data Device

Log Device

0 1 2 3 4
A B

Time
Write IO Logging and Backup with WalB

Time series of write I/Os

Data Device

0 1 2 3 4
A  B

Log Device

Write A'

A'  B

Scan the log device and generate a diff image
Write IO Logging and Backup with WalB

Time series of write I/Os

Data Device

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
</tbody>
</table>

Log Device

| Time | 1 | A' |   |   |   |

Scan the log device and generate a diff image
Performance test

- Compared dm-snap, dm-thin, and WalB
- Executed a workload during a backup
  - The workload & the backup will affect each other
- Measured the following metrics:
  - Latencies of the workload
  - Backup time
Environment & Settings

Test environment:

- CPU: 2.40 GHz x 12 cores
- MEM: 192 GiB
- HDD: 4 TB HDD, RAID 6 (8D2P)
- NIC: 10 Gbps x 2
- Kernel: 4.11 (latest upstream)

Test settings:

- 100 GiB volumes
- Workload: 4 KiB Random writes for a 5 GiB range
# Measuring the Backup Time (dm-snap, dm-thin)

## 4 KiB Random Writes

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5 GiB</td>
<td>95 GiB (unchanged)</td>
</tr>
</tbody>
</table>

- **dm-snap**: scan full image
- **dm-thin**: scan changed chunks (tree traversal)

**dm-snap**: take a snapshot & scan full image

**dm-thin**: get a structure of snapshot trees & find modified blocks & read these blocks
Measuring the Backup Time (WalB)

WalB: scan logs from a log device & send them to a backup server continuously

4 KiB Random Writes

| 5 GiB | 95 GiB (unchanged) |

WalB Device

Write IO logs

Log Device

Backup Server

Network

WalB: scan logs
Write I/O latency

IO spikes due to CoW, worse than dm-snap!

Small overhead due to CoW

WalB

no-backup

dm-snap

dm-thin

walb
Backup time

Backup taking time [sec]

- dm-snap: 1146 sec
- dm-thin: 2260 sec
- walb: 1.2 sec

so fast!
slower than dm-snap
Conclusion

- dm-snap & dm-thin
  - High I/O latency during a backup
  - Long backup time

- WalB
  - Stable and low I/O latency (no spikes)
  - Short backup time

WalB satisfies our requirements for production use.
Try WalB!

- Project page
  - https://walb-linux.github.io/

- Tutorial
  - Vagrantfile for Ubuntu 16.04 and CentOS 7
Q&A

email: kota-uchida@cybozu.co.jp

twitter: @uchan_nos