openstack JAPAN

openstack

Open source software to build public and private clouds.

Object Storage; Overview

2011.11.30

日本 OpenStack ユーザ会
Tomoaki Nakajima / @irix_jp

Agenda

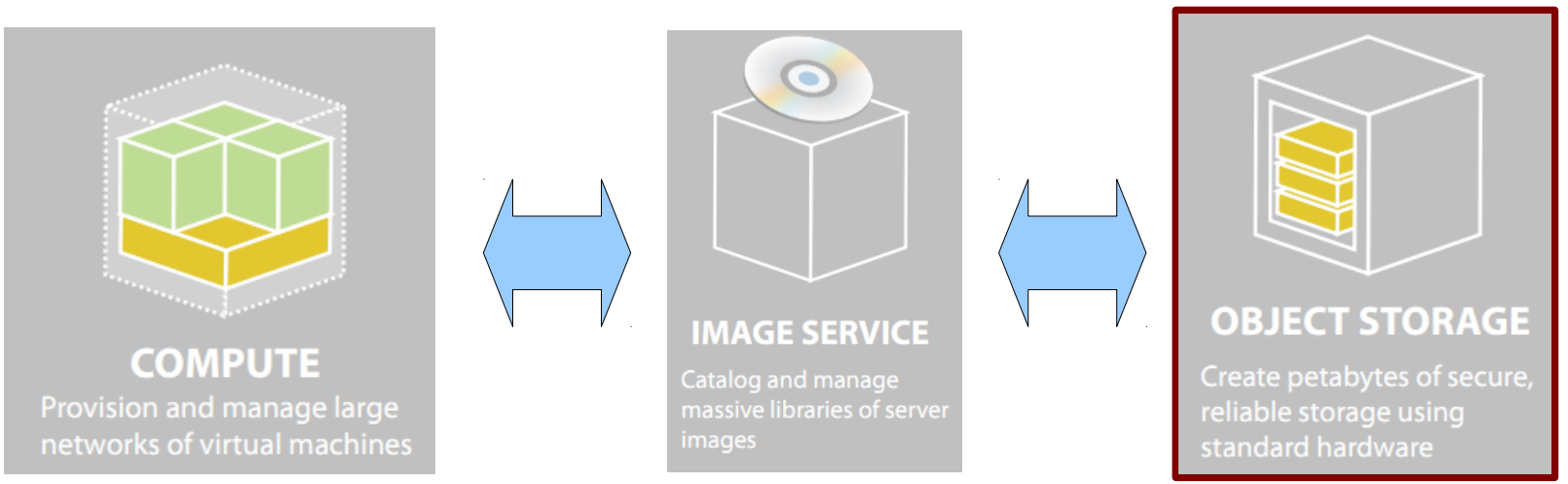
- Swift って何？
- Swift の利点
- Swift を使う
- Ring
- Ring 考察
- まとめ
- Swift ニュース
- お知らせ

＊ OSC2011TokyoFall で予告した「障害時の動作」については時間の都合で省略しました。

Swiftって何？

Swift って何？

- OpenStack の一部で Object Storage 機能を担当します。
- Amazon S3 相当です（互換 API あり）
- 普段は Glance と連携して、 Nova が使う仮想マシンイメージの保存先として動きます。

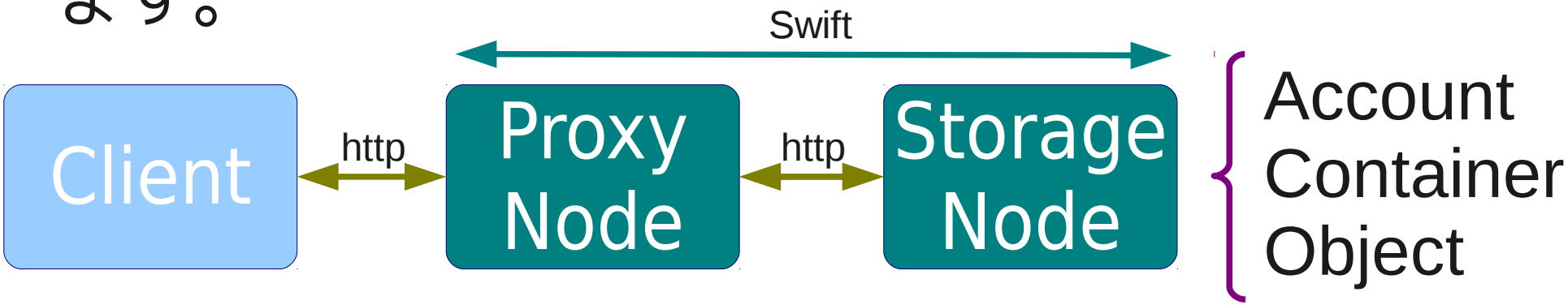


Swift って何？

- 普段は縁の下の力持ち的な存在ですが、果たしてその実態は・・・

Swift って何？

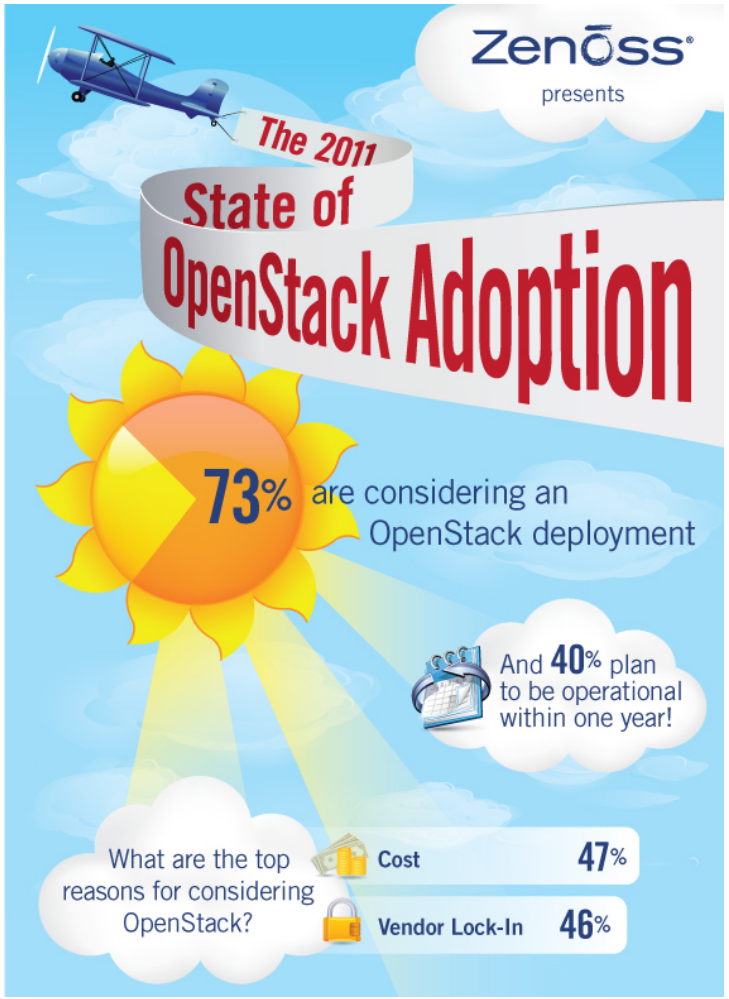
- その実態は単なるファイルサーバです
  - オブジェクト≒ファイル
- HTTP （ REST ）でオブジェクトの操作を行います。

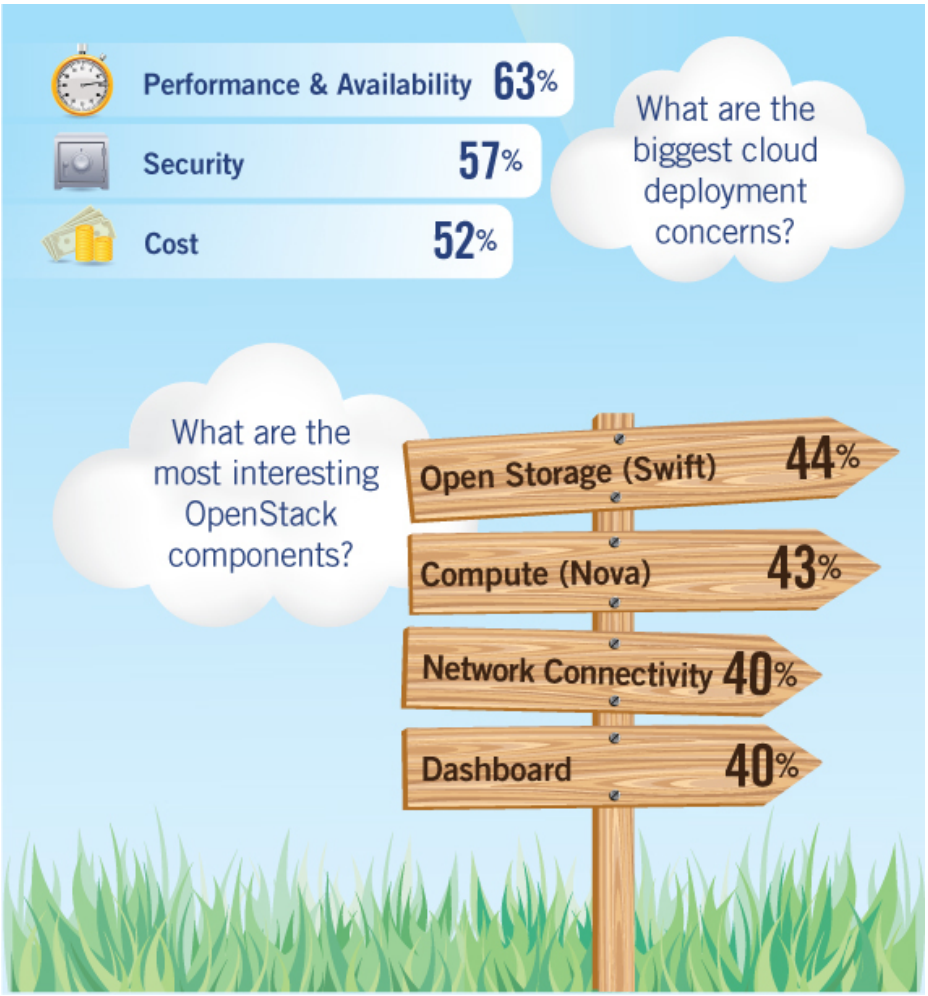


- 一見すると単なるファイルサーバーですが・・・

Swift って何？

- 実は OpenStack の人気者





http://blog.zenoss.com/2011/11/zenoss-2011-openstack-adoption-survey-infographic-and-results/

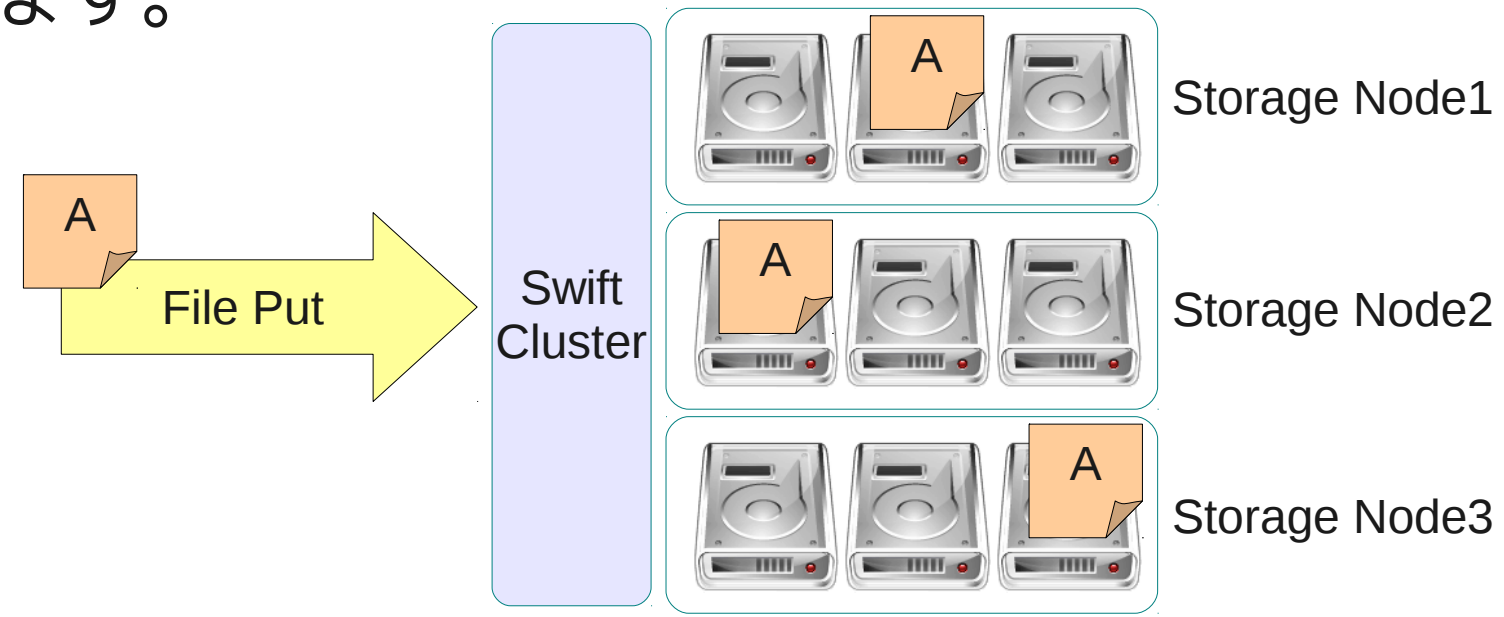
Swift って何？

- Swift の人気の秘密はいったい！？

Swiftの利点

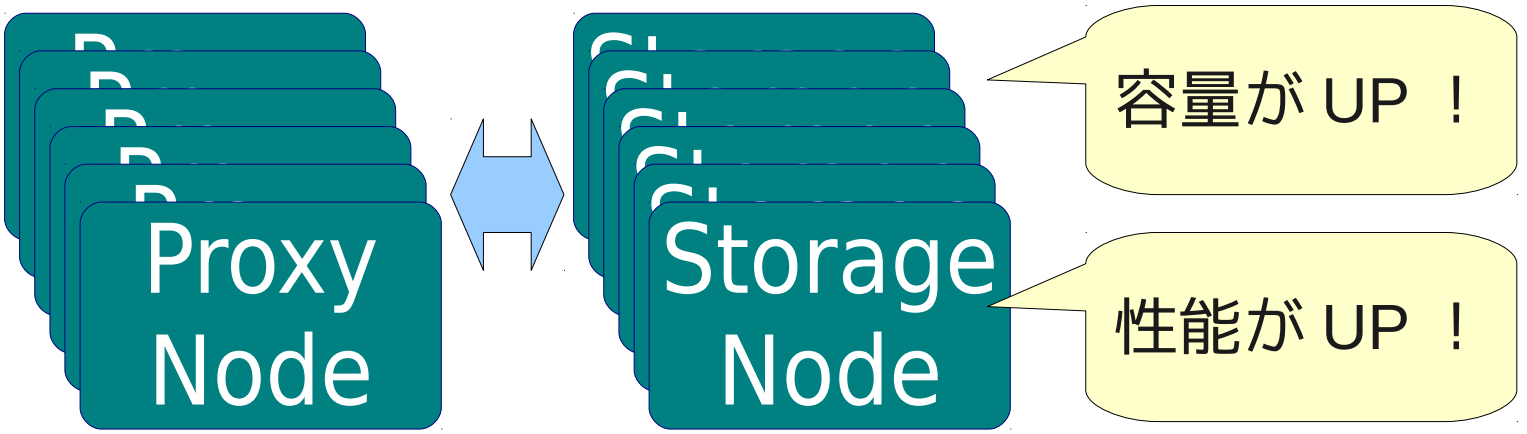
Swift って何がいいの？ #1

- 安価なハードウェアで安全にファイルの保存が可能です。
  - RAID も不要です。
  - デフォルトで１つのオブジェクトに３つの複製を作成します。



Swift って何がいいの？ #2

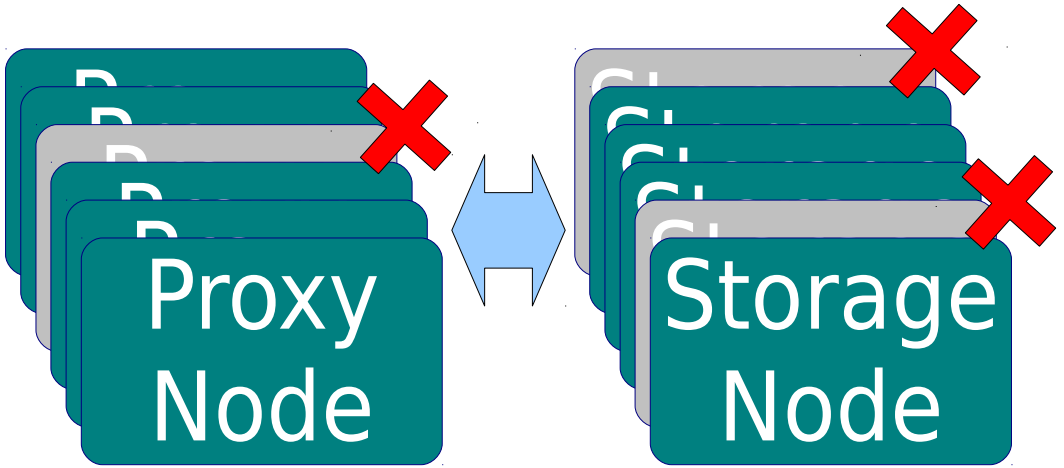
- Web サーバみたいに簡単にスケールします。



- Proxy Node へのアクセスは DNS ラウンドロビンか、ロードバランスで分配
- Storage Node は Proxy が振り分け

Swift って何がいいの？ #3

- 中央データベースや共有ストレージを持たないシンプルな構造なので・・・
  - スケール時のボトルネックが存在しません
    - 認証部分が怪しいようですが・・・
  - SPOF が存在しません



Swift って何がいいの？ #4

- 強力な自己修復機能を持っています。
  - 保存されたオブジェクトの破損や消失を自動検知して自動で修復してくれます。
  - HDD やノードごと吹っ飛んでも、 HW を復旧すれば周辺からデータが複製されて勝手に回復します。
  - 運用が楽チン！
    - 修復や複製には rsync が使われます。

Swift って何がいいの？ #5

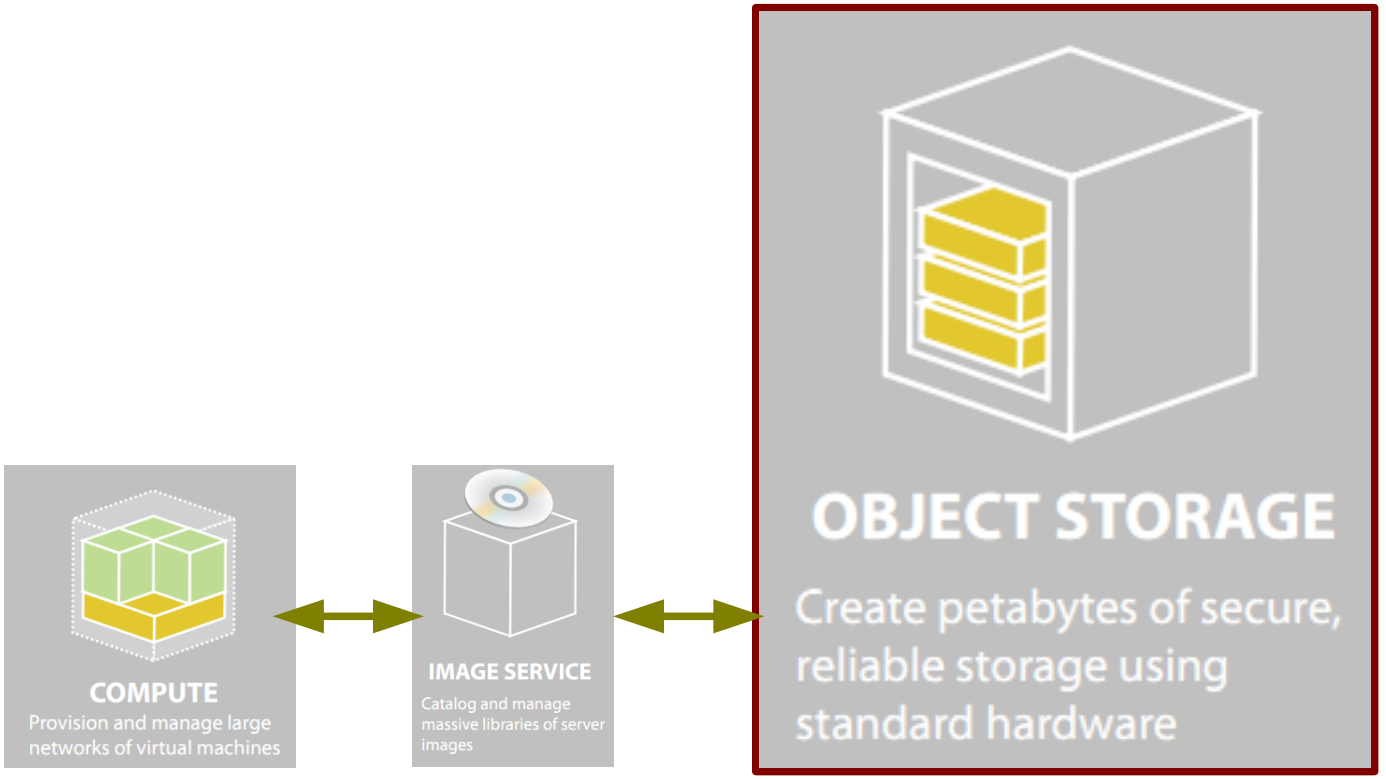
- その他
  - 商用サービス（ Rackspace Cloud Files ）をベースとしているので OpenStack の中でも相対的に完成度が高いです。
  - ファイルにいろんな拡張属性を与えられます
    - 単なるファイルの入れ物ではなく、ファイルと情報をセットにしたオブジェクトとして扱えます。
    - 将来的には拡張属性の検索機能が実装される予定です。

Swift って何がいいの？

- Swift の一番いいところ、それは・・・

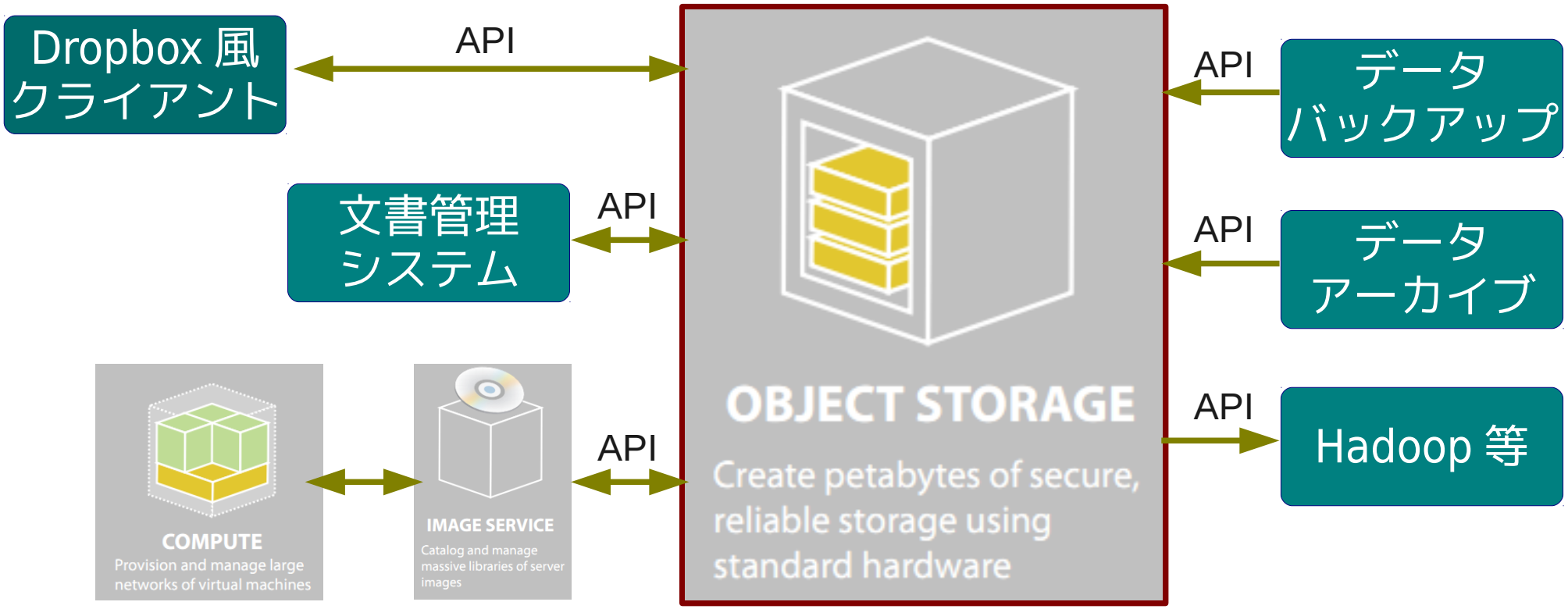
Swift って何がいいの？

- Swift の一番いいところ、それは・・・
  汎用性が高く『単体』でも使い道があります。



Swift って何がいいの？

- Swift の一番いいところ、それは・・・
  汎用性が高く『単体』でも使い道があります。



Swiftを使う

Swift を使う

- インストールや設定方法は省略
  - 「 OpenStack Swift install 」でぐぐる
    - OpenStack の大容量ストレージサービス、 Swift の使い方 - TechTarget
      - http://techtarget.itmedia.co.jp/tt/news/1109/20/news02.html
    - OpenStack Storage(Swift) 調査報告書
      - http://www.creationline.com/lab/772
    - SAIO - Swift All In One
      - http://swift.openstack.org/development_saio.html
    - Instructions for a Multiple Server Swift Installation
      - http://swift.openstack.org/howto_installmultinode.html

Swift を使う

- 大雑把な流れ（ RHEL 系）
  - 準備
    - rpm -ihv openstack-repo-2011.3-0.2.noarch.rpm
    - yum install openstack-swift-* xinetd rsync memcached
    - xfs ファイルシステムの作成 (xattr 対応 FS…ext3/4 も可)
  - 設定ファイルの編集
    - proxy-server.conf 、 account-server.conf
    - container-server.conf 、 object-server.conf
  - **Ring** の作成

Ring?

Ring

- Ring は Swift の根幹機能を制御する重要ファイル
  - データの分散配置
  - 分散したデータへのアクセス経路
  - レプリケーション

Ring

- ほぼ全ての機能が Ring を参照する
- そのため Swift クラスタに参加する全ノードに**同一 Ring を配布**しておく必要がある。
- Ring は**管理者が手動で編集**しないと変わらない。

Ring

- Ring を作成するために必要な情報
  - パーティション数
  - レプリカ数
  - ゾーン
  - ストレージノード IP
  - ストレージデバイス名
  - デバイス優先度

- swift-ring-builder コマンドで作成・編集

Ring

- Ring 作成の流れ
  - Create
    ↓
  - Add
    ↓
  - Rebalance
    ↓
  - 全 Swift ノードに配布
    - account, container, object
      それぞれに作成する必要あり。

Ring を作る

Create/Add/Rebalance

- 例) swift-ring-builder *Target* create 18 3 24
  - Target
    - account.builder
    - container.builder
    - object.builder
  - 18　・・・パーティション数
  - 3　・・・レプリケーション数 [個]
  - 24　・・・リバランス待ち時間 [h]

Ring を作る

swift-ring-builder *Target* create 18 3 24

- パーティション数
  - パーティションは Swift がデータを配置する単位
  - Swift クラスタ内に何個のパーティションを作成するか指定する。
    - マニュアルによる推奨値は
      - 最終的にクラスタ内に含めるディスク本数を１００倍
      - その数に最も近い 2^n を求める。
      - 求めた n をパーティション数として使用する。
        - 18 の場合、 2^18 = 262144 / 100 ≒ 2600 本
- レプリケーション数はそのままの意味、リバランス待ち時間については後で解説

Ring にデバイスを追加

Create/**Add**/Rebalance

- 例)swift-ring-builder *Target* add \

  z1-192.168.1.101:6000/disk1_meta 1000

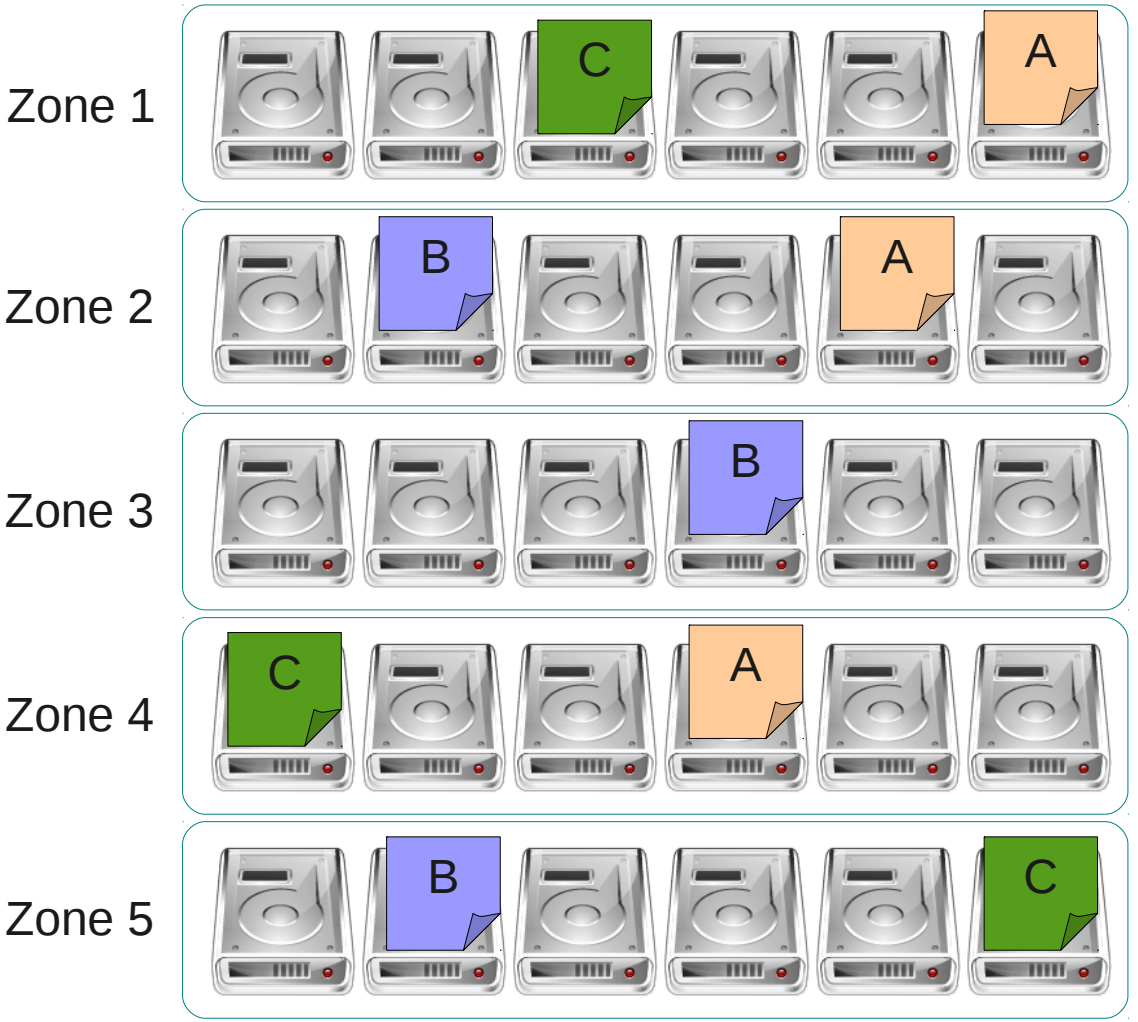
  - z1　　・・・ゾーン番号（ z2,z3,z4... ）
  - 192.168.1.101:6000

    ・・・ストレージノード IP/PORT
  - /disk1 ・・・デバイス
  - _meta ・・・エイリアス
  - 1000 ・・・優先度

Ring にデバイスを追加

z1-192.168.1.101:6000/disk1_meta 1000

- ゾーン番号
  - 複製の範囲を決定する
  - データの複製は必ず異なるゾーンに属するディスクへ行われる
  - 同一ゾーンに属するディスクへは複製されない



Ring にデバイスを追加

z1-**192.168.1.101:6000/disk1_meta 1000**

- IP/PORT、デバイス
  - そのまんま。 Swift で使うディスクを設定する。
    - 物理デバイス名ではなくマウントポイントを指定
- エイリアス
  - 上記の IP/PORT、デバイスの組合せに別名をつける。後で Ring を編集する場合にこのエイリアスを指定して操作できる。省略可能。
- 優先度
  - 相対値で指定。大きいものにデータが多く配置される。

Ring のリバランス

Create/Add/**Rebalance**

- 例)swift-ring-builder *Target* rebalance

  - 所属するディスクに優先度に基づくパーティションの割り当てを行う

Ring 構築の例

- Ring の作成とデバイスの追加例

```
swift-ring-builder obj.bldr create 18 3 24
swift-ring-builder obj.bldr add z1-1.1.1.1:100/disk1 100
swift-ring-builder obj.bldr add z2-2.2.2.2:100/disk2 100
swift-ring-builder obj.bldr add z3-3.3.3.3:100/disk3 100
swift-ring-builder obj.bldr add z4-4.4.4.4:100/disk4 100
swift-ring-builder obj.bldr add z5-5.5.5.5:100/disk5 100
```

- この状態でリバランスを行うと・・・

```
swift-ring-builder obj.bldr rebalance
```

obj.bldr ... object.builder の略

Ring 構築の例

- パーティションが割り当てられる

```
object.builder, build version 5
262144 partitions, 3 replicas, 5 zones, 5 devices, 0.00
balance
The minimum number of hours before a partition can be
reassigned is 24
Devices:
 id zone ip address port  name weight partitions balance meta
 0     1    1.1.1.1  100 disk1 100.00     157286   -0.00
 1     2    2.2.2.2  100 disk2 100.00     157287    0.00
 2     3    3.3.3.3  100 disk3 100.00     157286   -0.00
 3     4    4.4.4.4  100 disk4 100.00     157286   -0.00
 4     5    5.5.5.5  100 disk5 100.00     157287    0.00
```

Ring 構築の例

- パーティションが割り当

2^18 = 262,144

262,144 x 3 = 786,432

```
object.builder, build version 5
262144 partitions, 3 replicas, 5
balance
The minimum number of hours before a partition can be
reassi
Device
 id zo                                    partitions balance meta
 0      1     1.1.1.1  100 disk1 100.00      157286   -0.00
 1      2     2.2.2.2  100 disk2 100.00      157287    0.00
 2      3     3.3.3.3  100 disk3 100.00      157286   -0.00
 3      4     4.4.4.4  100 disk4 100.00      157286   -0.00
 4      5     5.5.5.5  100 disk5 100.00      157287    0.00
```

sum(partitions)
= 786,428

Ring まとめ

- Ring は Swift の重要ファイル
  - Swift は Ring でデータの配置を決定する

- swift-ring-builder コマンドを使用する
  - accout, container, object それぞれ作成する
  - create → add → rebalance → 配布の順
  - 全ノードに同一 Ring を配布する必要がある

Ring 考察

Ring の概要

A
GroupA/UserA
ContainerA
filenameA

B
GroupB/UserB
ContainerB
filenameB

swift_hash_path_suffix

Part1 Part2 Part3 Part4 ・・・ 2^n

Replica1 array('H', [0, 1, 5, 8, 7, 6, ...])
Replica2 array('H', [6, 3, 9, 4, 3, 2, ...])
Replica3 array('H', [4, 7, 2, 0, 5, 1, ...])

{... 'ip': '192.168.128.12', 'id': 4, 'parts': 10, 'device': 'disk5', 'port': 6000 ...},
{... 'ip': '192.168.128.12', 'id': 5, 'parts': 10, 'device': 'disk6', 'port': 6000 ...},
{... 'ip': '192.168.128.12', 'id': 6, 'parts': 10, 'device': 'disk7', 'port': 6000 ...},...

＊間違ってたらすみません。後でこっそり教えてください。

Ring のメリット

静的にデータ配置の決定

- 同じデータは同じ場所に配置される
- ファイルベースの分散
- 他ノードの状態を意識する必要がない

クラスタ内のノード間連結が弱い

- ボトルネックが発生しにくい
- 各ノードが同じ静的経路を持つので SPOF が無い
- 1 つの障害が全体に伝搬しにくい

構造がシンプルで
大規模ストレージ環境に適したモデル

Ring のデメリット

- 一度作った Ring のパーティション数とレプリカ数は変更できない
  - RingBuilder オブジェクトを直接いじれば・・・？
- 同じ手順で作った Ring に互換性が無い
  - Create → Add xxx → Rebalance = RingA
  - Create → Add xxx → Rebalance = RingB ≠ RingA

Ring のデメリット

- 同じ手順で作った Ring に互換性が無い
  - 初回リバランス時、デバイスをランダムにパーティションへ割り当てるため

```
'_replica2part2dev': [
array('H', [0, 1, 5, 8, 7, 6, 9, 8, 9, 8, 9, 3, ...]),
array('H', [6, 3, 9, 4, 3, 2, 3, 1, 2, 0, 6, 5, ...]),
array('H', [4, 7, 2, 0, 5, 1, 4, 7, 6, 5, 0, 8, ...])]
```

```
'_replica2part2dev': [
array('H', [8, 0, 1, 7, 5, 0, 6, 3, 2, 8, 5, 8, ...]),
array('H', [5, 2, 4, 4, 1, 7, 3, 1, 9, 4, 1, 7, ...]),
array('H', [3, 6, 9, 8, 2, 9, 0, 7, 6, 3, 9, 2, ...])]
```

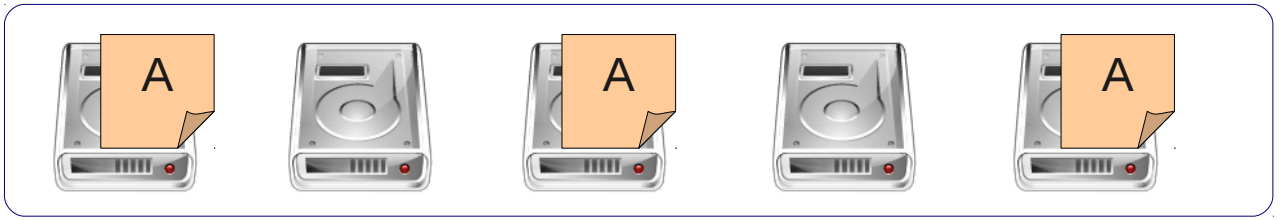
Ring の注意点

- バックアップ重要
  - Ring は編集すると同一ディレクトリにバックアップが作成される
    - backups/<unix_time>.target
  - 何重にもバックアップをとっておいて損はない
    - Swift に突っ込んでおくのもあり
- 編集は必ず既存 Ring に対して行う
  - Create → add xxx → rebalance
    → add/remove yyy → rebalance

Ring のメンテナンス

- 変更した Ring を配布中に新旧 Ring が混在する場合はどうなるのか？

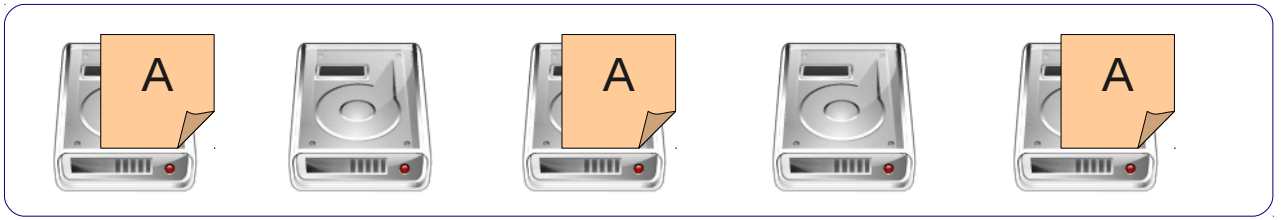
Proxy Node



Ring のメンテナンス

- このような場合

Proxy Node



追加

Ring のメンテナンス

- 新旧 Ring の混在は一時的には OK
  - 例）以下のような Ring を作成

```
swift-ring-builder object.builder create 8 3 1

swift-ring-builder object.builder add z1-1.1.1.1:111/disk1 100
swift-ring-builder object.builder add z2-1.1.1.1:111/disk2 100
swift-ring-builder object.builder add z3-1.1.1.1:111/disk3 100
swift-ring-builder object.builder add z4-1.1.1.1:111/disk4 100
swift-ring-builder object.builder add z5-1.1.1.1:111/disk5 100
swift-ring-builder object.builder rebalance
```

Ring のメンテナンス

- Ring の状態

```
256 partitions, 3 replicas, 5 zones, 5 devices, 0.39 balance
The minimum number of hours before a partition can be
reassigned is 1
Devices:
id zone ip address port  name weight partitions balance meta
0     1    1.1.1.1  111 disk1 100.00        154    0.26
1     2    1.1.1.1  111 disk2 100.00        153   -0.39
2     3    1.1.1.1  111 disk3 100.00        153   -0.39
3     4    1.1.1.1  111 disk4 100.00        154    0.26
4     5    1.1.1.1  111 disk5 100.00        154    0.26
```

Ring のメンテナンス

- Ring に優先度の大きいデバイスを追加

```
swift-ring-builder object.builder add z1-2.2.2.2:222/disk1 1000
swift-ring-builder object.builder add z2-2.2.2.2:222/disk2 1000
swift-ring-builder object.builder add z3-2.2.2.2:222/disk3 1000
swift-ring-builder object.builder add z4-2.2.2.2:222/disk4 1000
swift-ring-builder object.builder add z5-2.2.2.2:222/disk5 1000
```

Ring のメンテナンス

- Ring の状態（リバランス前）

```
id zone ip address port  name weight partitions balance meta
0     1    1.1.1.1  111 disk1  100.00        154 1002.86
1     2    1.1.1.1  111 disk2  100.00        153  995.70
2     3    1.1.1.1  111 disk3  100.00        153  995.70
3     4    1.1.1.1  111 disk4  100.00        154 1002.86
4     5    1.1.1.1  111 disk5  100.00        154 1002.86
5     1    2.2.2.2  222 disk1 1000.00          0 -100.00
6     2    2.2.2.2  222 disk2 1000.00          0 -100.00
7     3    2.2.2.2  222 disk3 1000.00          0 -100.00
8     4    2.2.2.2  222 disk4 1000.00          0 -100.00
9     5    2.2.2.2  222 disk5 1000.00          0 -100.00
```

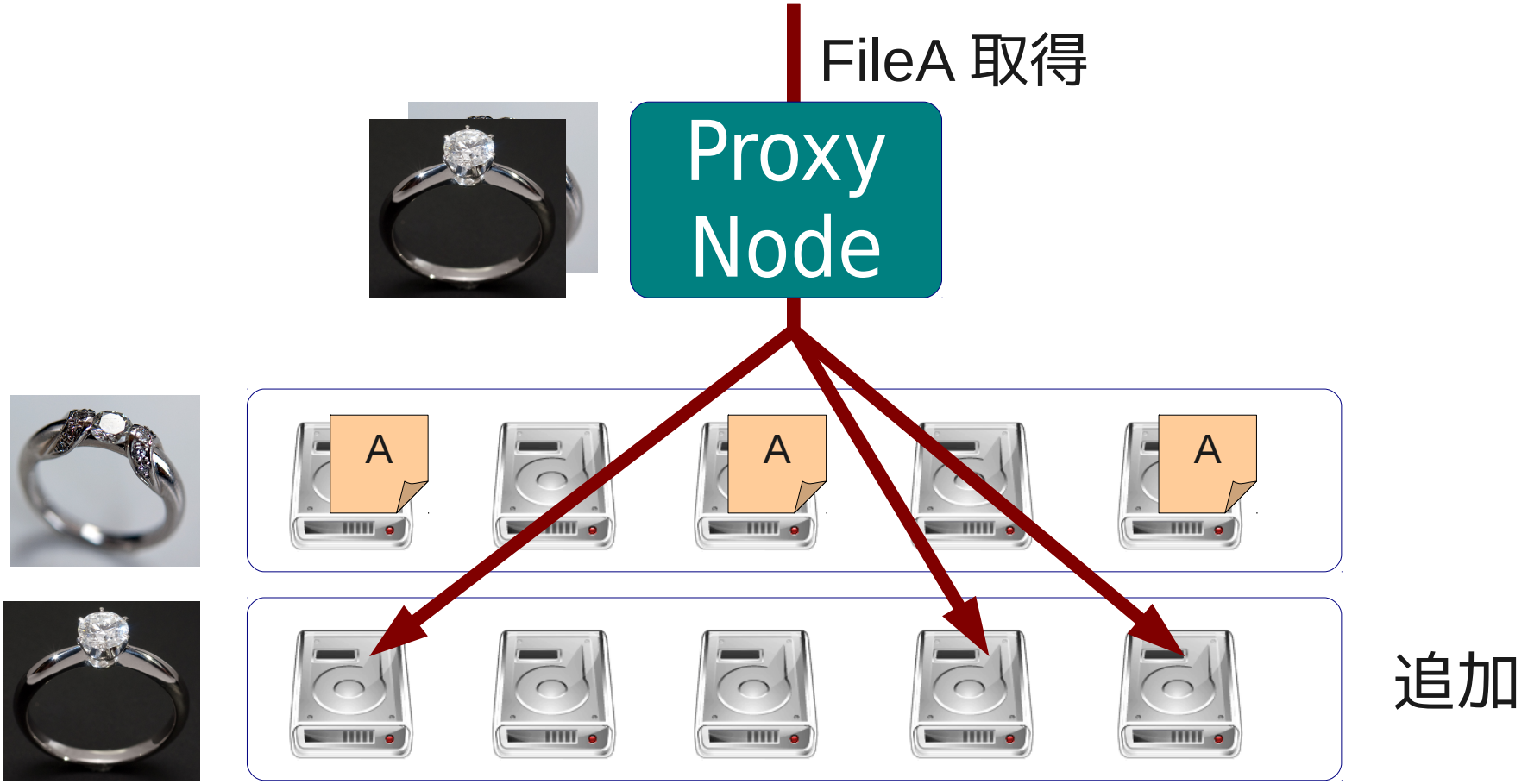
Ring のメンテナンス

- このまま Rebalance が実行されてしまうと...

```
id zone ip address port  name weight partitions balance meta
0     1    1.1.1.1  111 disk1  100.00        154 1002.86
1     2    1.1.1.1  111 disk2  100.00        153  995.70
2     3    1.1.1.1  111 disk3  100.00        153  995.70
3     4    1.1.1.1  111 disk4  100.00        154 1002.86
4     5    1.1.1.1  111 disk5  100.00        154 1002.86
5     1    2.2.2.2  222 disk1 1000.00          0 -100.00
6     2    2.2.2.2  222 disk2 1000.00          0 -100.00
7     3    2.2.2.2  222 disk3 1000.00          0 -100.00
8     4    2.2.2.2  222 disk4 1000.00          0 -100.00
9     5    2.2.2.2  222 disk5 1000.00          0 -100.00
```

Ring のメンテナンス

- 大幅なパーティション移動が起こりデータへのアクセスができなってしまう？？



Ring のメンテナンス

- 実際はそうはならず・・

初期

id	zone	ip	name	weight	parts
0	1	1.1.1.1	disk1	100.00	154
1	2	1.1.1.1	disk2	100.00	153
2	3	1.1.1.1	disk3	100.00	153
3	4	1.1.1.1	disk4	100.00	154
4	5	1.1.1.1	disk5	100.00	154
5	1	2.2.2.2	disk1	1000.00	0
6	2	2.2.2.2	disk2	1000.00	0
7	3	2.2.2.2	disk3	1000.00	0
8	4	2.2.2.2	disk4	1000.00	0
9	5	2.2.2.2	disk5	1000.00	0

Ring のメンテナンス

- 徐々に・・・

						初期	1 回目
id	zone	ip	name	weight	parts		parts
0	1	1.1.1.1	disk1	100.00	154		99
1	2	1.1.1.1	disk2	100.00	153		102
2	3	1.1.1.1	disk3	100.00	153		97
3	4	1.1.1.1	disk4	100.00	154		112
4	5	1.1.1.1	disk5	100.00	154		102
5	1	2.2.2.2	disk1	1000.00	0		51
6	2	2.2.2.2	disk2	1000.00	0		51
7	3	2.2.2.2	disk3	1000.00	0		52
8	4	2.2.2.2	disk4	1000.00	0		51
9	5	2.2.2.2	disk5	1000.00	0		51

Ring のメンテナンス

- 徐々に分散され・・・

id	zone	ip	name	weight	parts (初期)	parts (1 回目)	parts (2 回目)
0	1	1.1.1.1	disk1	100.00	154	99	58
1	2	1.1.1.1	disk2	100.00	153	102	56
2	3	1.1.1.1	disk3	100.00	153	97	48
3	4	1.1.1.1	disk4	100.00	154	112	55
4	5	1.1.1.1	disk5	100.00	154	102	39
5	1	2.2.2.2	disk1	1000.00	0	51	102
6	2	2.2.2.2	disk2	1000.00	0	51	103
7	3	2.2.2.2	disk3	1000.00	0	52	103
8	4	2.2.2.2	disk4	1000.00	0	51	102
9	5	2.2.2.2	disk5	1000.00	0	51	102

Ring のメンテナンス

- 徐々に分散されます

id	zone	ip	name	weight	parts 初期	parts 1 回目	parts 2 回目	parts 3 回目
0	1	1.1.1.1	disk1	100.00	154	99	58	14
1	2	1.1.1.1	disk2	100.00	153	102	56	13
2	3	1.1.1.1	disk3	100.00	153	97	48	14
3	4	1.1.1.1	disk4	100.00	154	112	55	14
4	5	1.1.1.1	disk5	100.00	154	102	39	14
5	1	2.2.2.2	disk1	1000.00	0	51	102	140
6	2	2.2.2.2	disk2	1000.00	0	51	103	140
7	3	2.2.2.2	disk3	1000.00	0	52	103	140
8	4	2.2.2.2	disk4	1000.00	0	51	102	139
9	5	2.2.2.2	disk5	1000.00	0	51	102	140

Ring のメンテナンス

- Ring のバランスが変動する場合はレプリカ単位で Rebanace が実行される。
  - 急激なパーティションの移動によるデータ消失・アクセス不可を防ぐための仕組み。
  - 複数回の RebaInace が必要になる。この時、運用上の安全のため、次の Rebalance までロックをかけるのが「リバランス待ち時間」
    - Swift-ring-builder xxx create 18 3 24
  - １つのレプリカしか移動しないので、残りのレプリカを使ってデータへアクセスできる。

Ring のメンテナンス

- レプリカ単位での Rebalance

初期

```
array('H', [2, 1, 1, 4, 0, 2, 3, 3, 4, 4, 0, 2, 4, ...]),
array('H', [0, 3, 3, 3, 4, 0, 4, 2, 3, 0, 3, 4, 1, ...]),
array('H', [4, 0, 2, 2, 1, 1, 0, 1, 2, 1, 1, 0, 3, ...])
```

Ring のメンテナンス

- レプリカ単位での Rebalance

初期

```
array('H', [2, 1, 1, 4, 0, 2, 3, 3, 4, 4, 0, 2, 4, ...]),
array('H', [0, 3, 3, 3, 4, 0, 4, 2, 3, 0, 3, 4, 1, ...]),
array('H', [4, 0, 2, 2, 1, 1, 0, 1, 2, 1, 1, 0, 3, ...])
```

1 回目

```
array('H', [8, 9, 6, 6, 8, 9, 7, 5, 9, 7, 9, 8, 9, ...]),
array('H', [0, 3, 3, 3, 4, 0, 4, 2, 3, 0, 3, 4, 1, ...]),
array('H', [4, 0, 2, 2, 1, 1, 0, 1, 2, 1, 1, 0, 3, ...])
```

Ring のメンテナンス

- レプリカ単位での Rebalance

初期

```
array('H', [2, 1, 1, 4, 0, 2, 3, 3, 4, 4, 0, 2, 4, ...]),
array('H', [0, 3, 3, 3, 4, 0, 4, 2, 3, 0, 3, 4, 1, ...]),
array('H', [4, 0, 2, 2, 1, 1, 0, 1, 2, 1, 1, 0, 3, ...])
```

1 回目

```
array('H', [8, 9, 6, 6, 8, 9, 7, 5, 9, 7, 9, 8, 9, ...]),
array('H', [0, 3, 3, 3, 4, 0, 4, 2, 3, 0, 3, 4, 1, ...]),
array('H', [4, 0, 2, 2, 1, 1, 0, 1, 2, 1, 1, 0, 3, ...])
```

2 回目

```
array('H', [8, 9, 6, 6, 8, 9, 7, 5, 9, 7, 9, 8, 9, ...]),
array('H', [7, 6, 5, 8, 9, 7, 8, 9, 6, 8, 7, 9, 6, ...]),
array('H', [4, 0, 2, 2, 1, 1, 0, 1, 2, 1, 1, 0, 3, ...])
```

Ring のメンテナンス

- レプリカ単位での Rebalance

初期

```
array('H', [2, 1, 1, 4, 0, 2, 3, 3, 4, 4, 0, 2, 4, ...]),
array('H', [0, 3, 3, 3, 4, 0, 4, 2, 3, 0, 3, 4, 1, ...]),
array('H', [4, 0, 2, 2, 1, 1, 0, 1, 2, 1, 1, 0, 3, ...])
```

1 回目

```
array('H', [8, 9, 6, 6, 8, 9, 7, 5, 9, 7, 9, 8, 9, ...]),
array('H', [0, 3, 3, 3, 4, 0, 4, 2, 3, 0, 3, 4, 1, ...]),
array('H', [4, 0, 2, 2, 1, 1, 0, 1, 2, 1, 1, 0, 3, ...])
```

2 回目

```
array('H', [8, 9, 6, 6, 8, 9, 7, 5, 9, 7, 9, 8, 9, ...]),
array('H', [7, 6, 5, 8, 9, 7, 8, 9, 6, 8, 7, 9, 6, ...]),
array('H', [4, 0, 2, 2, 1, 1, 0, 1, 2, 1, 1, 0, 3, ...])
```

3 回目

```
array('H', [8, 9, 6, 6, 8, 9, 7, 5, 9, 7, 9, 8, 9, ...]),
array('H', [7, 6, 5, 8, 9, 7, 8, 9, 6, 8, 7, 9, 6, ...]),
array('H', [5, 8, 8, 7, 7, 6, 9, 7, 2, 6, 5, 6, 5, ...])
```

Ring のメンテナンス

- 編集した Ring を配布するときに Swift が止まるんじゃ？
  - Reload オプションがあるが・・・
  - 接続済みのセッションは通信が終わるまで待ってるっぽいが・・・？？
  - 大丈夫な気もしますが要確認（これから
    - 知ってる方、後で教えてください！

Ring の今後

- 全ノードに配布って・・・チョーいけてない！
  - ノード追加、 HDD 追加
    - その都度全ノードに配布・リロードする必要がある。
  - Essex では ring-builder-server が実装される予定
    - Web ベースでの Ring 作成・編集
    - ノードへの配布機能
    - https://blueprints.launchpad.net/swift/+spec/ring-builder-ser
  - ちょっとは楽になりそう？？

まとめ

まとめ

- Swift は・・・
  - HTTP （ REST ）で通信するファイルサーバです。
  - 安価なハードで安全に動きます。 RAID 不要。
  - 容量・性能がスケールし、 SPOF がありません。
  - 勝手に自己修復します。
  - 汎用性が高く Swift 単体で使えます。
  - シンプルで大規模環境に適したストレージ構造
  - Ring 重要
    - 忘れずにバックアップしておきましょう

2011-11-24 Swift 1.4.4(Essex)

- Self Destructing Files
- add more detail to rate limit errors
- add swift man pages
- better ring builder error messages
- change ring builder exit codes
- create swift recon docs
- swift recon socket stats
- tempauth autocreate accounts
- zone specific recon

2011-11-24 Swift 1.4.4(Essex)

- Self Destructing Files
  - swift-object-expirer
  - 時間経過によるファイルの自動削除
  - アップロードしたオブジェクトに以下のメタデータを付与
    - X-Delete-At: 日時指定 (Unix 時間)
    - X-Delete-After: アップロードされてからの経過時間 (秒)
  - object-server.conf に以下を設定

```
[object-expirer]
interval = 300
```

お知らせ

- Swift 実証実験参加メンバー募集中
  - 大規模環境での Swift の運用を行い、問題点や注意点の洗い出しを計画しています。
  - 現在のステータス
    - 検証・検討内容の募集
    - HW、場所の募集 ← 重要！

  - クラウドストレージ研究会 ML にて議論中
    - http://groups.google.com/group/cloudstf
    - どしどしご参加ください！

ご静聴ありがとうございました。

参考サイト

- OpenStack(本家)
  - http://www.openstack.org/
- 日本 OpenStack ユーザ会
  - http://openstack.jp/
- API マニュアル（本家）
  - http://docs.openstack.org/api/openstack-object-storage/1.0/con

参考サイト

- 使用させていただいた素材
  - http://cool-liberty.com/
  - http://tanukifont.sblo.jp/article/41432838.html
  - http://butsudori.blog47.fc2.com/blog-entry-3.html
  - http://www.ashinari.com/

おまけ

自己修復機能

- ファイルが消えた
  - 1個・・・修復可能
  - 2個・・・修復可能
  - 3個・・・修復不可
- ファイルがストレージノードから消えても複製があればアクセス可能
- replicator で自動修復される

自己修復機能

- ファイルが壊れた
  - 1 個・・・修復可能
  - 2 個・・・修復可能
  - 3 個・・・修復不可
  - 壊れたファイルは quarantine ディレクトリへ移動される（ lost+found みたいなもの）
- ファイル破損は auditor により検知・除去され、 replicator により修復される

自己修復機能

- ファイルの整合性は拡張属性として付与

```
0000   80 02 7D 71 01 28 55 0E 43 6F 6E 74 65 6E 74 2D    ..}q.(U.Content-
0010   4C 65 6E 67 74 68 71 02 55 04 31 34 37 39 71 03    Lengthq.U.1479q.
0020   55 04 6E 61 6D 65 71 04 55 24 2F 41 55 54 48 5F    U.nameq.U$/AUTH_
0030   73 79 73 74 65 6D 2F 66 6F 6C 64 65 72 31 2F 61    system/folder1/a
0040   6E 61 63 6F 6E 64 61 2D 6B 73 2E 63 66 67 71 05    naconda-ks.cfgq.
0050   55 13 58 2D 4F 62 6A 65 63 74 2D 4D 65 74 61 2D    U.X-Object-Meta-
0060   4D 74 69 6D 65 71 06 55 0D 31 33 32 31 38 38 33    Mtimeq.U.1321883
0070   30 30 37 2E 30 32 71 07 55 04 45 54 61 67 71 08    007.02q.U.ETagq.
0080   55 20 35 62 38 38 63 35 34 61 34 35 62 34 64 65    U 5b88c54a45b4de
0090   33 37 62 61 32 34 30 38 62 32 66 65 33 38 37 39    37ba2408b2fe3879
00A0   30 36 71 09 55 0B 58 2D 54 69 6D 65 73 74 61 6D    06q.U.X-Timestam
00B0   70 71 0A 55 10 31 33 32 32 32 38 34 35 30 39 2E    pq.U.1322284509.
00C0   36 37 35 39 33 71 0B 55 0C 43 6F 6E 74 65 6E 74    67593q.U.Content
00D0   2D 54 79 70 65 71 0C 55 18 61 70 70 6C 69 63 61    -Typeq.U.applica
00E0   74 69 6F 6E 2F 6F 63 74 65 74 2D 73 74 72 65 61    tion/octet-strea
00F0   6D 71 0D 75 2E                                     mq.u.
```

好きな属性のセットも可能

```
> HEAD /v1/AUTH_system/folder1/anaconda-ks.cfg HTTP/1.1
> Host: 192.168.128.12:8080
> Accept: */*
> X-Auth-Token: AUTH_tk90678650dcde455cbf1a28f7f825cde2

< HTTP/1.1 200 OK
< Last-Modified: Thu, 24 Nov 2011 15:07:05 GMT
< Etag: 5b88c54a45b4de37ba2408b2fe387906
< X-Object-Meta-Installenv: Scientific Linux 6.1 x64
< X-Object-Meta-FileOwner: root
< X-Object-Meta-FileGroup: root
< X-Object-Meta-Permission: 0655
< Accept-Ranges: bytes
< Content-Length: 1479
< Content-Type: application/octet-stream
< Date: Sat, 26 Nov 2011 01:35:45 GMT
```

Swift API

- API マニュアル
  - OpenStack Object Storage Developer Guide
    - http://docs.openstack.org/api/openstack-object-storage/1.0/