GlusterFS紹介

OSS基盤技術センター
OSS技術第2課

GlusterFSとは？

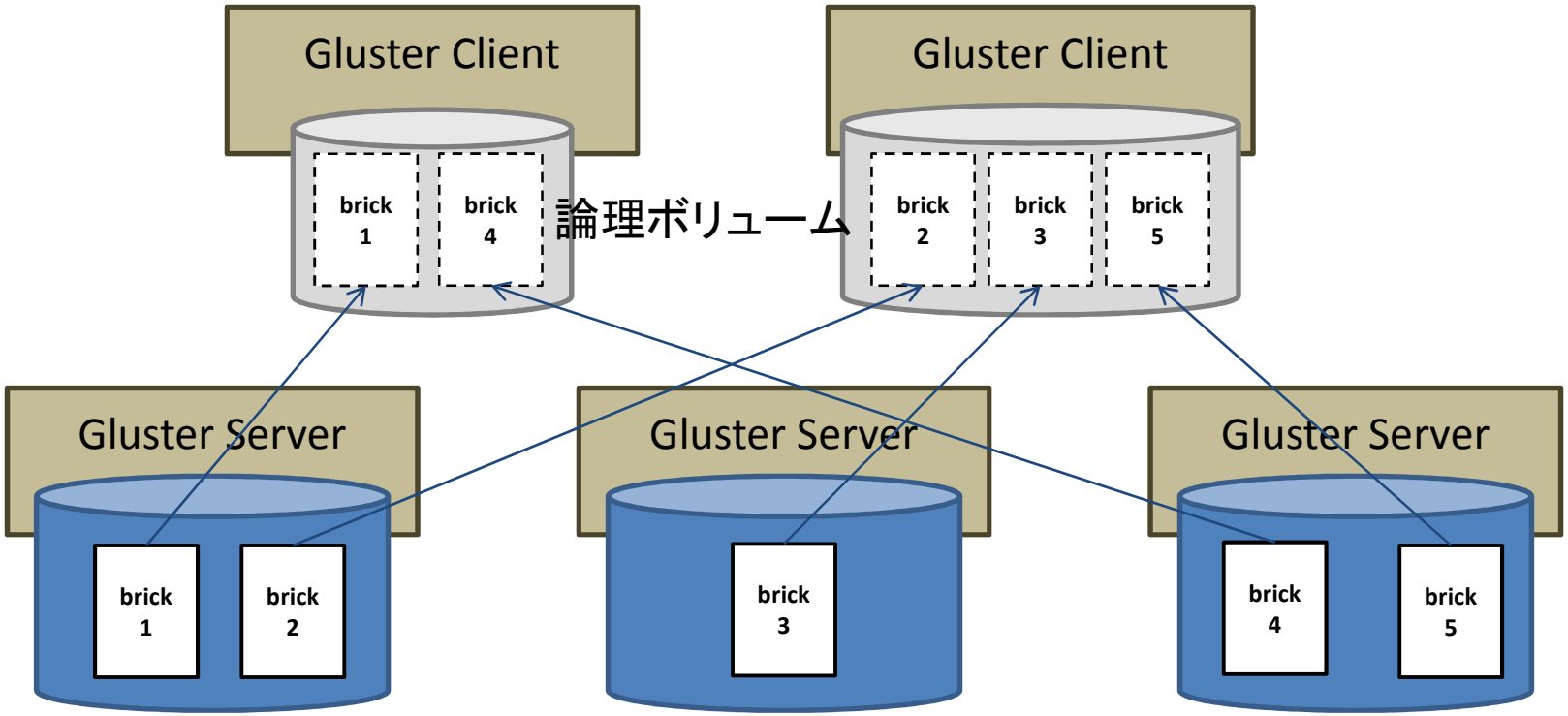
- Linux上で動作する分散ファイルシステム
  - Linuxのアプリケーションとして実装
- コモディティハードウェアのみで構成
- スケーラブル
- サーバの動的追加・削除が可能
- データ冗長化機能
- 負荷分散機能

歴史

- Gluster Inc. にて開発（インド バンガロール)
- 2011年9月Redhat社が買収
- 2013年3月現在、OSS版と商用版が存在する
  - コミュニティサイト http://www.gluster.org/
  - OSS最新版は、GlusterFS 3.3.1

Glusterfsの構成（1）

- Glusterサーバのローカルディスク上にBrickを配置
  - Brickの正体は、ローカルディスク上のファイルシステム内のディレクトリ
- Glusterクライアントが参照する論理ボリュームは、複数のBrickを束ねることにより生成
  - Brickの束ね方により、様々なタイプのボリュームを生成



Glusterfsの構成（2）

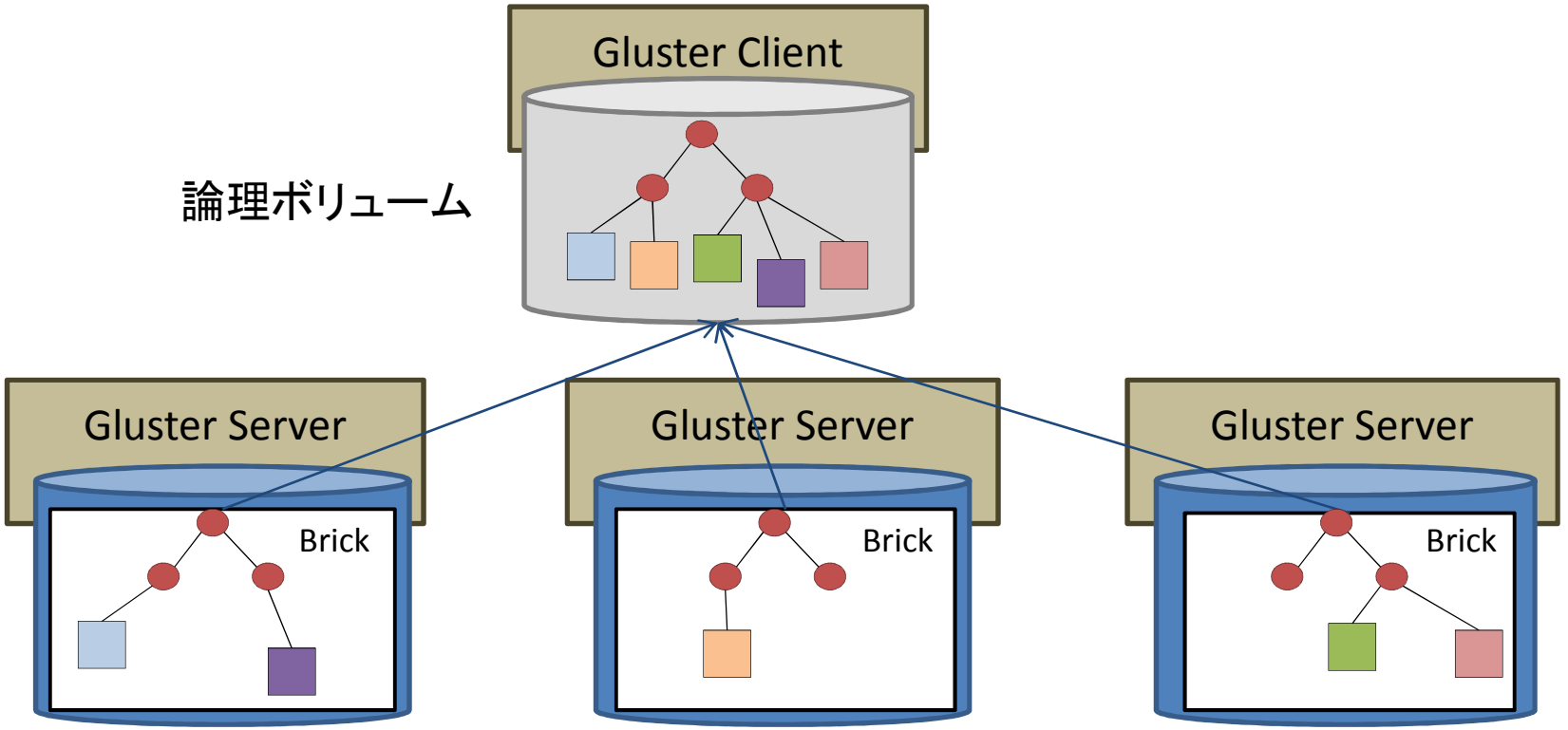
- メタデータノードなし
- ステートレス
- データの管理単位はファイル
  - ファイル毎に、配置するBrickを決定する
- 論理ボリュームとBrickの対応
  - 手動設定 （自動モードは無い）
    論理ボリューム作成時に、システム管理者が明示的にBrickの組み合わせを指定
    管理者は、すべてのサーバ、空きディスク容量、ネットワーク構成の情報から、最適なBrickの組み合わせを決める

論理ボリュームのタイプ

1. Distributedボリューム（分散ボリューム）
2. Replicatedボリューム（冗長ボリューム）
3. Stripedボリューム
4. タイプ1,2,3ボリュームの組み合わせ
   - 分散+冗長ボリュームなど

分散ボリューム（1）

- ファイルごとに、格納するBrickを選択する
  - ファイル名から計算したハッシュ値により、対応するBrickが一意に決定する
  - Swift (OpenStack Object Storage)と同じ戦略
- 全Brick内のファイルシステムを重ね合わせて、論理ボリューム内のファイルシステムとして見せる

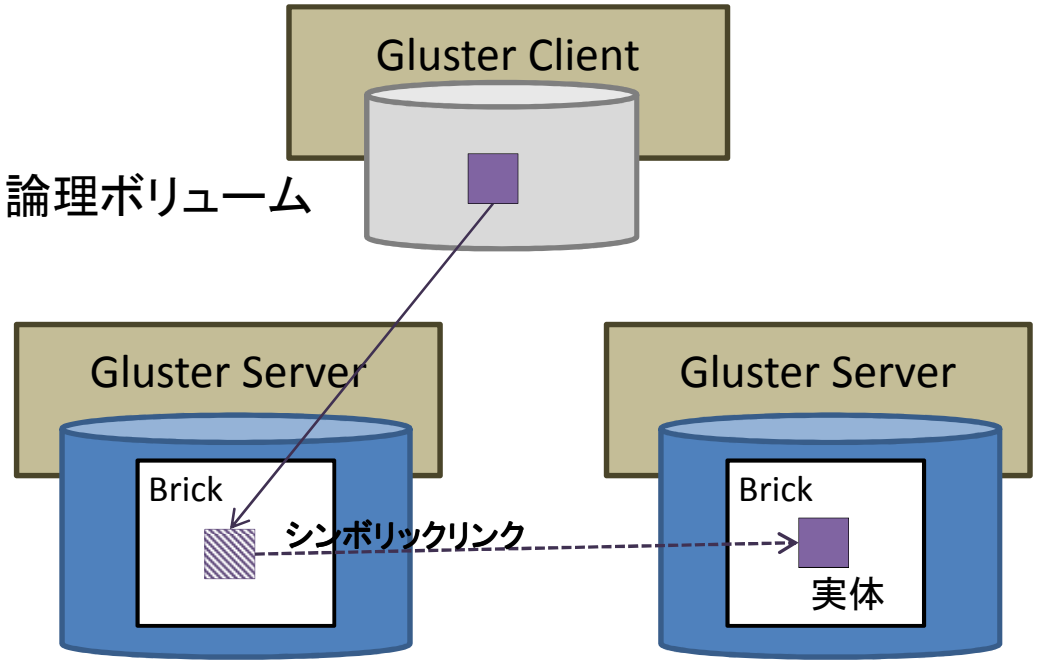


分散ボリューム（2）

- 分散ハッシュテーブル(DHT)
  - ファイル名とBrickの対応を管理するテーブル
    - ファイル名から計算したハッシュ値から、Brickを求めるためのテーブル
  - GlusterFS上の各ディレクトリがDHTを持つ
  - DHT毎にハッシュ値とBrickの対応が異なる
    - 特定のBrickにファイル格納が偏りにくくする
  - DHTは拡張アトリビュートに格納
    - Glusterサーバは、Brick用にXFS、ext4などの拡張アトリビュート対応のファイルシステムを利用する必要がある
- Brickのボリュームへの動的追加・削除可能

分散ボリューム（3）

- ファイル名の変更（rename）
  - 新ファイル名のハッシュ値が、今までと異なるBrickを指すようになる
  - ファイル実体は移動せず、ハッシュ値が指すBrickにシンボリックリンクファイルを置く
    - 拡張アトリビュートに、シンボリックリンク情報を格納

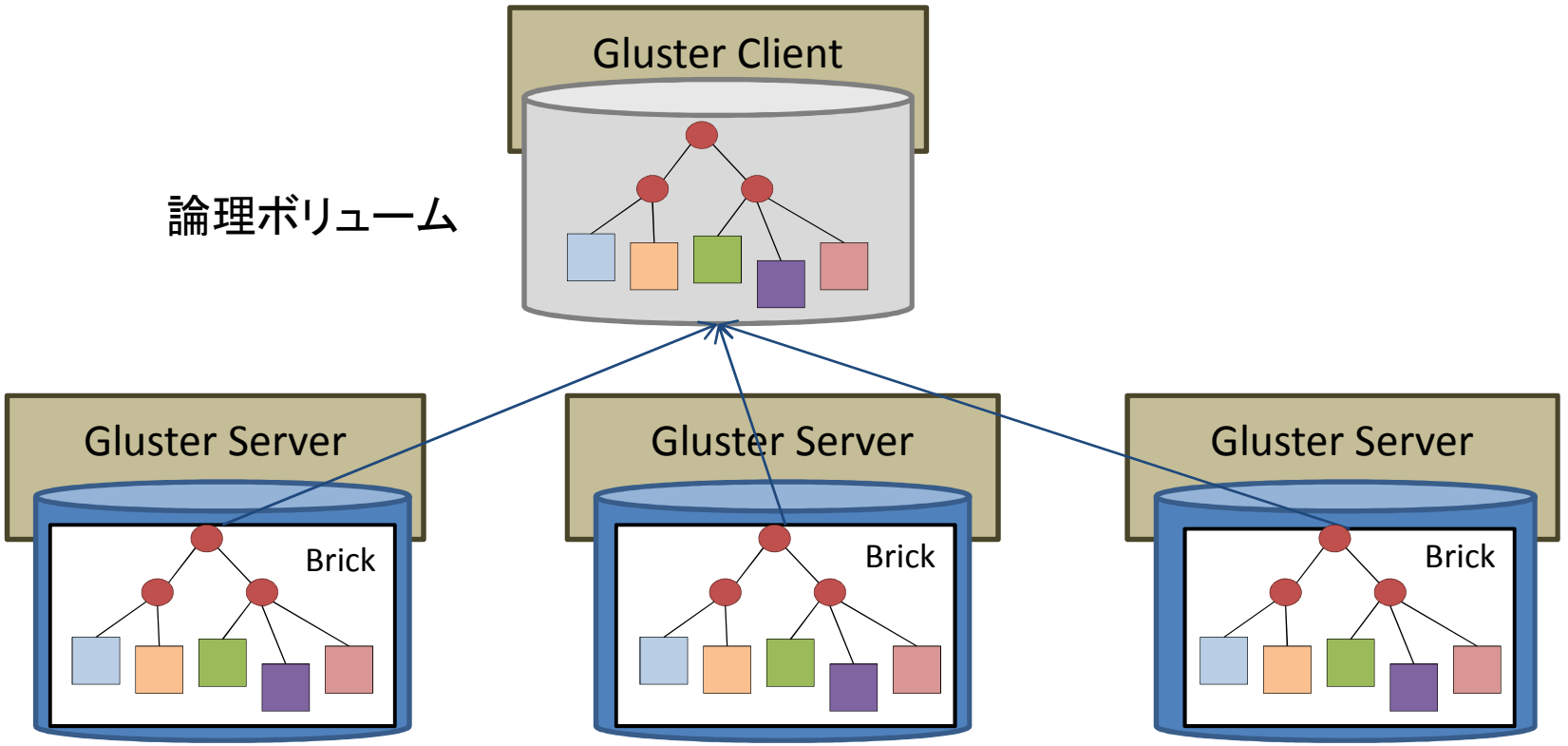


分散ボリューム（4）

- 運用中の論理ボリュームに新規Brickを追加可能
  - リバランス：既存ファイルの再配置(Brick間移動)は、明示的に指示する
  - リバランス中は、クライアントから論理ボリュームへのアクセスは不可
- サーバがダウンしてもクライアントは動作可能
  - ダウン中サーバが管理するBrick内に置かれたファイルは、クライアントから参照できない

冗長ボリューム（1）

- ファイル単位でミラー。論理ボリュームを構成するBrick すべてに、GlusterFS上のファイルの複製を持たせる
- ファイル更新時は、全てのBrick内のファイルを更新
- ファイル参照時は、任意のBrick内のファイルを参照
- データ冗長度は任意



冗長ボリューム（2）

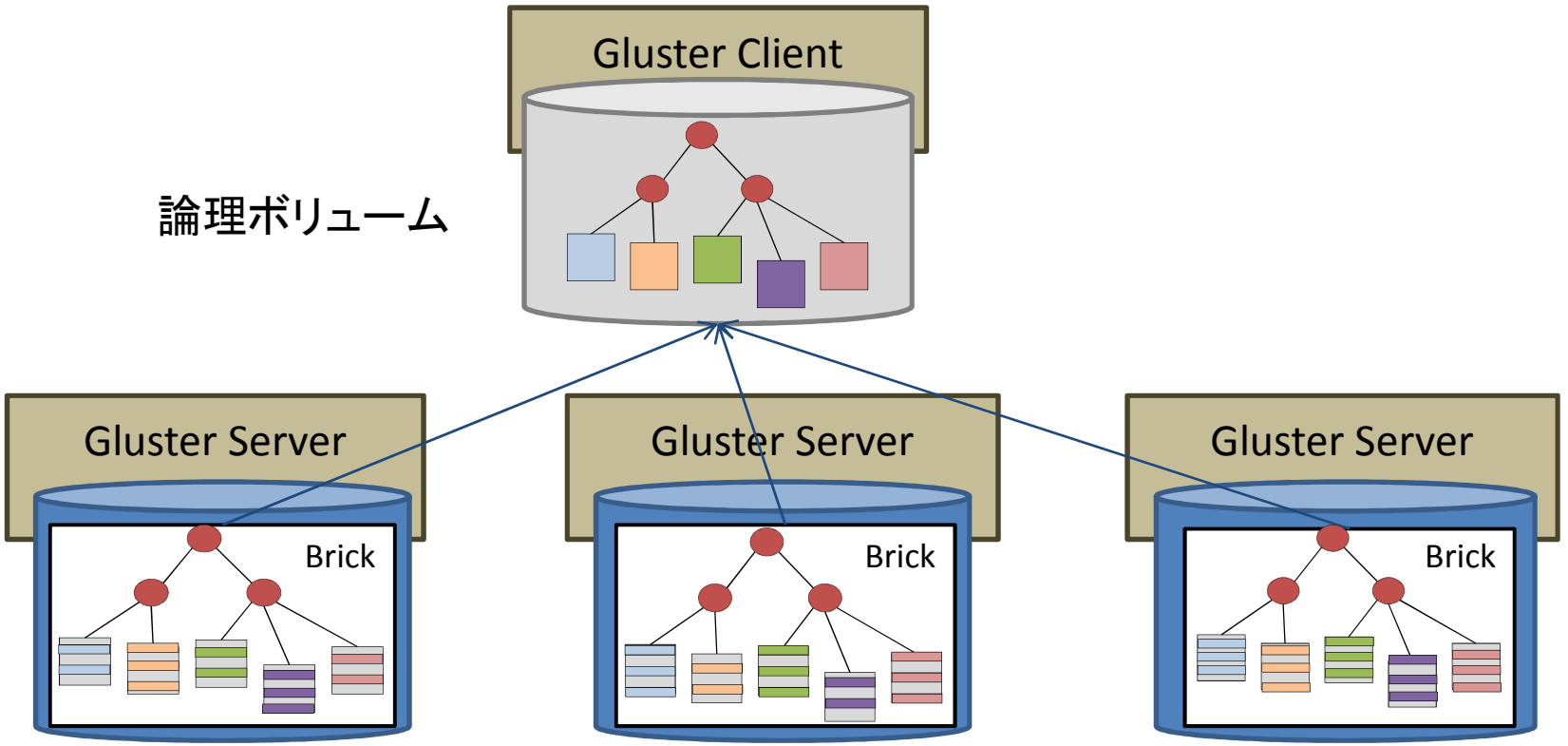
- 障害検出と修復
  - GlusterFS 3.3.1では、自動的に検出し修復が行われる（古いGlusterFSにはこの機能はない）
  - ディレクトリを参照した時、その配下のファイルの修復が行われる
  - 追加ファイル、更新ファイル、ファイル削除の状態の同期が行われる
  - サーバが故障から復帰するまで、論理ボリュームの修復は行われない
    - 論理ボリュームは冗長度不足の状態におかれる

冗長ボリューム（3）

- 冗長度の縮退中も、通常にファイルの読み書き、生成・削除が可能
- 冗長度の動的変更が可能

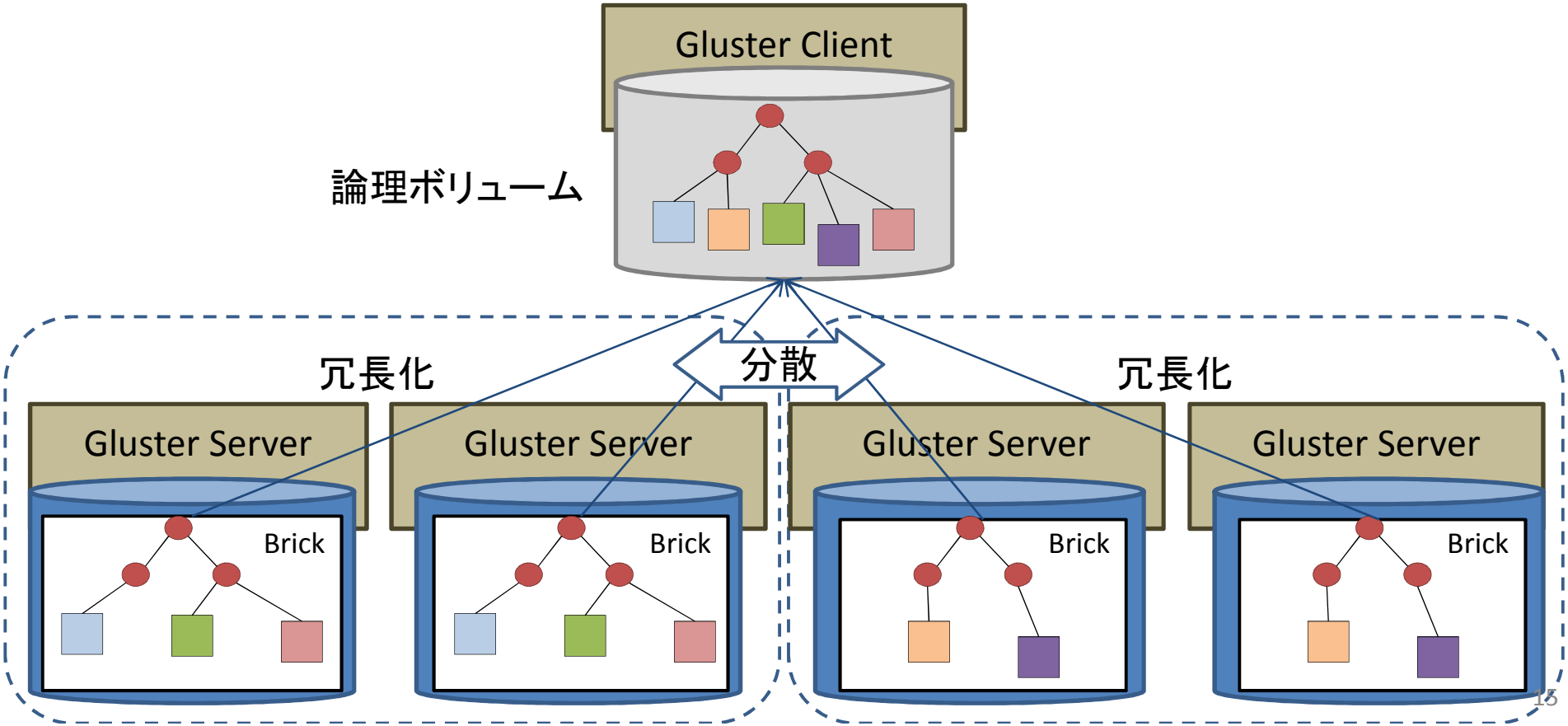
stripedボリューム（1）

- 1ファイルのデータを複数Brickに分散して配置する
- 1ファイルに対して複数クライアントからのアクセスがある場合、スループットが向上する



分散＋冗長ボリューム

- 複数の冗長ボリュームを束ねた分散ボリュームを作成することが可能
  - 全ての冗長ボリュームの冗長度は同じ
- 分散、冗長、stripedボリュームの任意の組み合わせが可能



クライアントAPI

- Glusterネイティブ
  - GlusterFSを直接マウントする。fuse機能を利用して実現
- NFS v3
  - 全Glusterサーバ上で、独自実装のNFSサーバが動作
  - lockd対応 (GlusterFS 3.3以降)
- Hadoop HDFS互換API
  - MapReduce処理のバックエンドストレージとして、HDFSの代わりに利用可能
- Swift互換 ReST API
- CIFS
  - Glusterクライアント上でSambaを動作させることにより実現する

パッケージ入手先

- RHEL6、CentOS6用のrpmパッケージ
  2013年3月末時の最新版は GlusterFS 3.3.1
  - http://download.gluster.org/pub/gluster/glusterfs/3.3/3.3.1/RHEL/epel-6
  - http://download.gluster.org/pub/gluster/glusterfs/3.3/3.3.1/CentOS/epel-6
  - epelレポジトリにあるものは少しバージョンが古い（ GlusterFS 3.2.7）
    - http://dl.fedoraproject.org/pub/epel/6/x86_64/repoview/glusterfs.html

インストールパッケージ

1. glusterfs-3.3.1-1.el6.x86_64.rpm
2. glusterfs-fuse-3.3.1-1.el6.x86_64.rpm
3. glusterfs-server-3.3.1-1.el6.x86_64.rpm
4. glusterfs-geo-replication-3.3.1-1.el6.x86_64.rpm
5. glusterfs-rdma-3.3.1-1.el6.x86_64.rpm
6. glusterfs-debuginfo-3.3.1-1.el6.x86_64.rpm
7. glusterfs-devel-3.3.1-1.el6.x86_64.rpm

サーバインストール（１）

- Glusterサーバ
  - パッケージ1, 2, 3 をインストール
  - パッケージ4, 5 は、利用する機能によってインストール
    - パッケージ4は、遠隔レプリケーション機能を利用する場合にインストール
    - パッケージ5は、RDMA可能なネットワーク(Infini Bandなど)を利用するときにインストール
- LinuxネイティブNFSサーバは止めておくこと

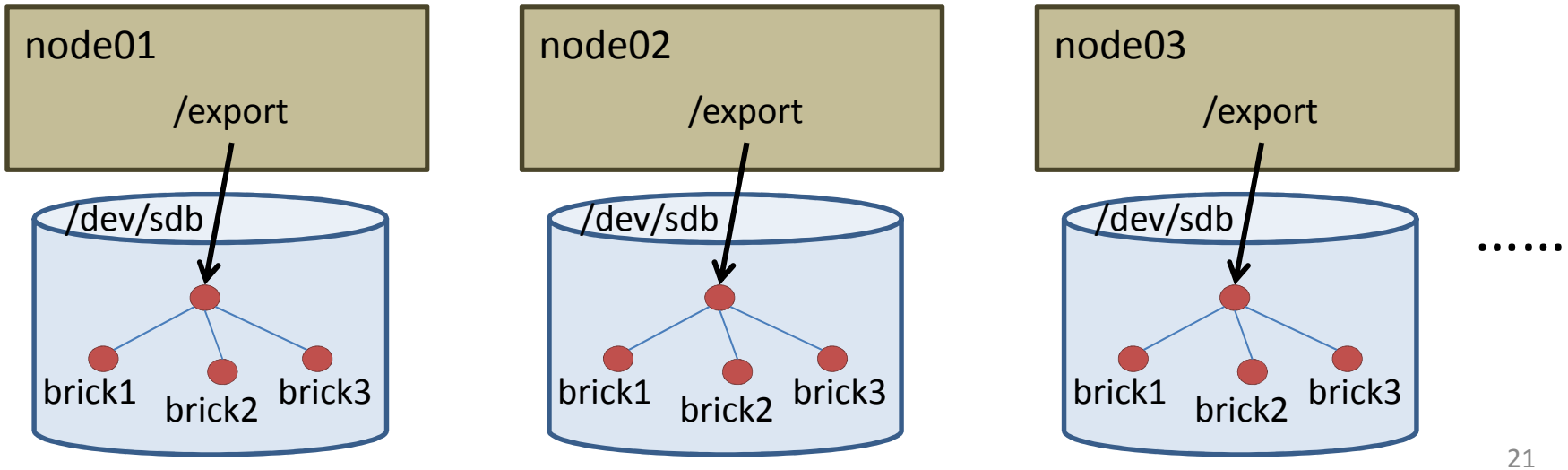
サーバインストール（2）

- アクセス許可するポート

ポート番号	説明
tcp 24007	Glusterが内部通信に利用
tcp 24008	Infinibandを使用する場合に使用
tcp 24009-240**	Brick管理用。Brick 1つに付き1ポート必要。24009～2409 + (Brick数 – 1)の範囲のポートを使用する。
tcp 38465-38467	Gluster NFS用
tcp 111	ポートマッパ用
udp 111	ポートマッパ用

設定例

- サーバ4台(node01, node02, node03, node04)から構成されるGluserFSシステムを構築
  - /dev/sdb を/exportにマウント、brick格納域とする
  - 例では（単純化のため）全サーバを同じ構成としているが、異なっていても構わない



ストレージプールの定義（1）

- GlusterFSを構成するGlusterサーバをプール(Trusted Storage Pool)に登録する
  - 任意のGlusterサーバ上で実行可能

例）Glusterサーバ3台(node01, node02, node03)をプールに登録する
  - node01上で、node02とnode03の登録処理を実行

```
[node01]# gluster peer probe node02
Probe successful
[node01]# gluster peer probe node03
Probe successful
```

Brickの用意（1）

- Brickを配置するファイルシステムは、拡張アトリビュート対応しているもの
  - XFSまたはext4
  - iノードのサイズは、拡張アトリビュート格納用に大きくする(512バイト)
  - ext4利用時は、明示的にmountオプションで拡張アトリビュート利用を指定(**user_xattr**オプション)

```
[node01]# mkfs.ext4 -I 512 /dev/sdb

[node01]# vi /etc/fstab
          :
/dev/sdb        /export        ext4    defaults,user_xattr 1 2
          :
```

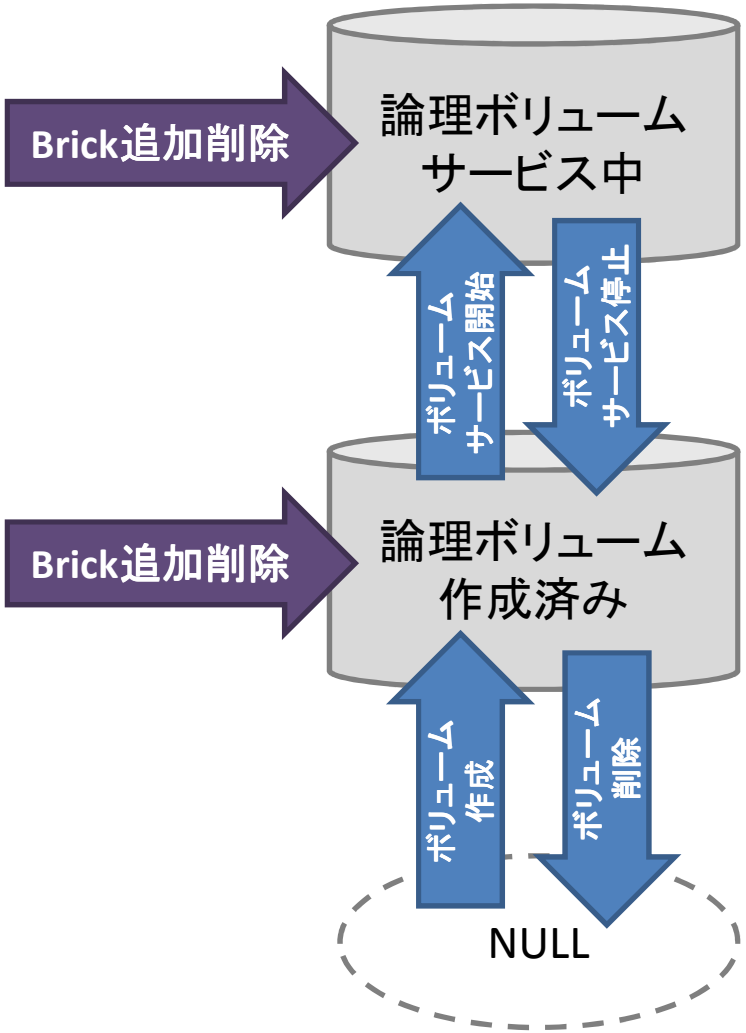
Brickの用意（2）

- Brick用のディレクトリを作成する

例）各Glusterサーバにて、Brick用ディレクトリ /export/brick1、/export/brick2、/export/brick3 を作成する

```
[node01]# mkdir /export/brick1 /export/brick2 /export/brick3
```

論理ボリュームの状態遷移



クライアントからマウント可能

論理ボリュームの作成とサービス開始

- node01とnode02上の2つのBrickから成る分散ボリュームvol1を作成
  - 任意のGlusterサーバ上で実行
- 論理ボリューム作成とボリュームサービス開始は別操作

```
[node01]# gluster volume create vol1 node01:/export/brick1
node02:/export/brick1
Creation of volume vol1 has been successful. Please start the
volume to access data.

[node01]# gluster volume start vol1
Starting volume vol1 has been successful
```

Glusterクライアントからの利用

- Glusterクライアントから、論理ボリュームをマウント
  - 任意のGlusterサーバを指定して、論理ボリュームvol1を/mntにマウント
    - ファイルシステムタイプglusterfsを指定
- マウント後は、通常のファイルシステムと区別なくアクセス可能

```
[client]# mount -t glusterfs node01:/vol1 /mnt
```

NFSクライアントからの利用

- 標準のLinux NFSクライアント機能を利用
- Glusterfsパッケージのインストール不要
- 任意のGlusterサーバ上のNFSサーバを指定してmountする
- プロトコルバージョンは、NFS v3
- NFS over TCP/NFS over UDPどちらも利用可能

例）論理ボリュームvol1をNFSマウントする

```
[client]# mount -t nfs -o vers=3 node01:/vol1 /mnt
```

論理ボリュームの削除

- 論理ボリュームvol1を削除
  - 任意のGlusterサーバ上で実行
  - 論理ボリュームvol1のサービスを停止させたのち、削除を行う

```
[node01]# gluster volume stop vol1
Stopping volume vol1 has been successful

[node01]# gluster volume delete vol1
Deleting volume will erase all information about the volume. Do
you want to continue? (y/n) y
Deleting volume vol1 has been successful
```

Glusterサーバの削除

- Glusterサーバをプール(Trusted Storage Pool)から削除する

例）Glusterサーバ(node02)をプールから削除

– node01上で、node02の削除処理を実行

```
[node01]# gluster peer detach node02
Detach successful
```

分散ボリューム – 作成

- 特にボリュームタイプを指定せずに論理ボリュームを作成すると、分散ボリュームになる
  - 論理ボリュームに参加させたいBrickすべてを、引数に指定する

```
[node01]# gluster volume create vol1 node01:/export/brick1
node02:/export/brick1
Creation of volume vol1 has been successful. Please start the
volume to access data.

[node01]# gluster volume start vol1
Starting volume vol1 has been successful
```

分散ボリューム - Brick内データ構造

- ディレクトリ
  - 論理ボリューム上に作成されたディレクトリは、全てのBrick内に同じ名前で作成される
- 通常ファイル
  - 論理ボリューム上に作成されたファイルは、Brickのひとつの中に同名のファイルとして作成される
- .glusterfsディレクトリ
  - GlusterFS管理用のデータが置かれる
- DHT (分散ハッシュテーブル）
  - 各ディレクトリの拡張アトリビュート領域に格納

分散ボリューム – 障害時の動作（1）

- 論理ボリュームを構成するBrickを抱えるサーバのひとつがダウン
  - 動作しているサーバ上のBrick内のファイルは参照可能
  - 初回アクセス時、サーバダウン検出が確定するまでの時間(40秒程度）ブロックされる

分散ボリューム – 障害時の動作（２）

- 論理ボリュームを構成するBrickを抱えるサーバのひとつがダウン
  - ファイルの作成、書き込み、renameは、故障しているBrickを参照する操作の場合、エラーとなる
  - ディレクトリ作成は、常にエラーとなる

分散ボリューム – 拡張（1）

- 論理ボリュームvol1に、Brick /export/brick1を追加

```
[node01]# gluster volume add-brick vol1 node03:/export/brick1
Add Brick successful
```

分散ボリューム – 拡張(2)

- 新規作成したディレクトリ配下のファイルの配置は、全Brick間でバランスされる
- 既存ディレクトリ配下のファイルは、追加したBrick内に自動的には移動しない
  - DHTは、ディレクトリ作成時に初期化される
    既存ディレクトリのDHTには、新規に追加したBrickの情報は登録されていない
  - 明示的にリバランス処理を指示する必要がある
- リバランス処理中は、論理ボリュームへのアクセスはブロックされる
- 必要性が無ければ(低ければ)、リバランス処理は行わなくてよい

分散ボリューム – 拡張(3)

- リバランス処理の開始

```
[node01]# gluster volume rebalance vol1 start
Starting rebalance on volume vol1 has been successful
```

- リバランス処理の監視
  - リバランス処理の進行状況を確認することができる

```
[node02]# gluster volume rebalance vol1 status
Node Rebalanced-files size scanned failures status
--------- ----------------- ----- ---------- --------  --------
node01     5     3.2MB     120       0    in progress
node02     9     2.3MB     408       0    completed
node03     3     1.0MB     276       2    in progress
```

分散ボリューム – Brickの交換

- メンテナンスなどのためにBrickを交換することができる
  - 手順は、Brick削除と同じ

```
[node02]# gluster volume replace-brick vol1
node02:/export/brick1 node04:/export/brick7 start

[node02]# gluster volume replace-brick vol1
node02:/export/brick1 node04 status

[node02]# gluster volume replace-brick vol1
node02:/export/brick1 node04 commit
```

冗長ボリューム－ 作成（1）

- ボリューム作成時に、冗長度を指定する
- 冗長度数分のBrickを指定

例）冗長度2の論理ボリュームvol2を作成

```
[node01]# gluster volume create vol2 replica 2
node01:/export/brick2 node02:/export/brick2
Creation of volume vol2 has been successful. Please start the
volume to access data.

[node01]# gluster volume start vol2
Starting volume vol2 has been successful
```

冗長ボリューム – ファイル作成（1）

- 作成したファイルの複製が全brickに置かれることを確認する

```
[client]# touch /mnt/foo/{hoge1,hoge2}

[client]# touch /mnt/bar/{hoge3,hoge4,hoge5}

[client]# ls /mnt/foo /mnt/bar
/mnt/bar:
hoge3  hoge4  hoge5

/export/brick2/foo:
hoge1  hoge2
```

冗長ボリューム – 冗長度変更

- 論理ボリュームに追加するBrickを、データの冗長度をあげるために利用する

例）冗長度2の論理ボリュームvol2に、Brickを一つ追加し、冗長度3の論理ボリュームに変更する

```
[node01]# gluster volume add-brick vol2 replica 3
node03:/export/brick3
Add Brick successful
```

注意: 追加ボリューム内のファイルの複製処理開始は、実際に利用するときまで遅延される

分散冗長ボリューム – 作成（1）

- 冗長ボリューム作成時に、冗長度の倍数のBrickを登録する

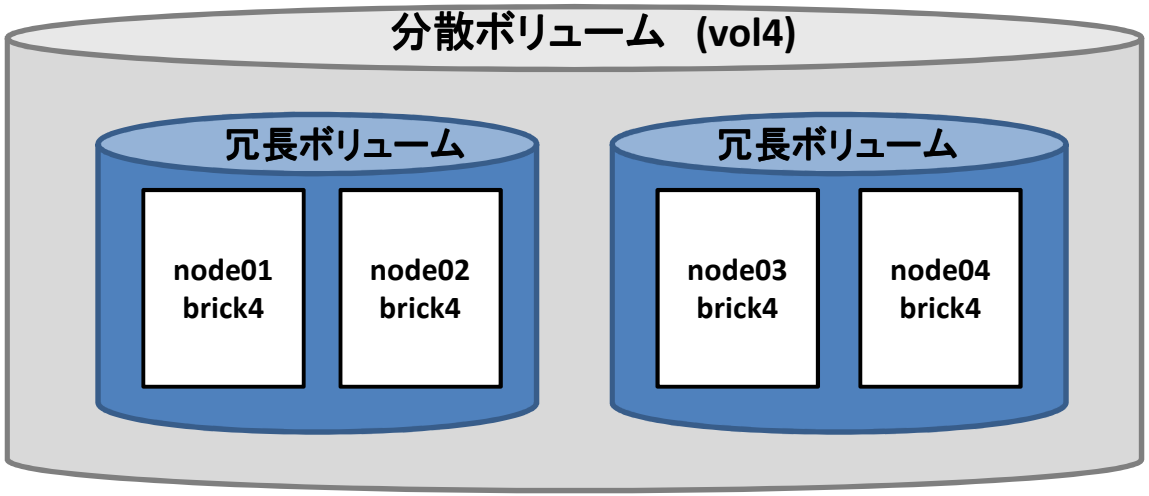
例）冗長度2、分散数2の論理ボリュームvol4を作成

```
[node01]# gluster volume create vol4 replica 2
node01:/export/brick4 node02:/export/brick4
node03:/export/brick4 node04:/export/brick4
Creation of volume vol4 has been successful. Please start the
volume to access data.

[node01]# gluster volume start vol4
Starting volume vol4 has been successful
```

分散冗長ボリューム（2）

- replicaオプションで指定した数ごとにBrickを組み合わせて、冗長ボリュームを作成
- 全ての冗長ボリュームを組み合わせて、分散ボリュームとする



stripedボリューム – 作成

- ボリューム作成時に、ストライプ数を指定する
- ストライプ数分のBrickを指定

注) まだ、開発段階にある機能

例）ストライプ数3の論理ボリュームvol3を作成

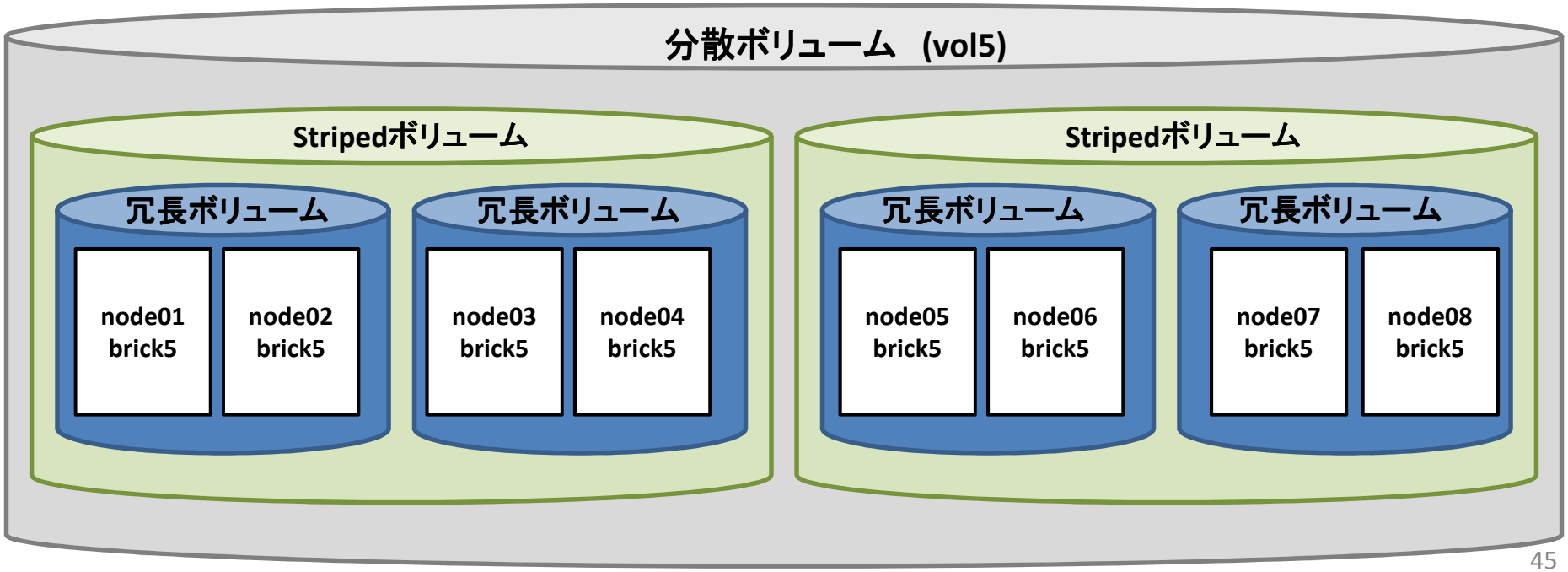
```
[node01]# gluster volume create vol3 stripe 3 node01:/export/brick3
node02:/export/brick3 node03:/export/brick3
Creation of volume vol3 has been successful. Please start the volume
to access data.

[node01]# gluster volume start vol3
Starting volume vol3 has been successful
```

Glusterサーバ上の各ファイルは、スパースファイルとして作成される

分散冗長stripedボリューム

- replicaオプションで指定した数ごとにBrickを組み合わせて、冗長ボリュームを作成
- stripeオプションで指定した数ごとに冗長ボリュームを組み合わせて、stripedボリュームを作成
- 全てのstripedボリュームを組み合わせて、分散ボリュームとする



RDMAの利用

- ノード間通信にRDMAを利用させることが可能
- 論理ボリュームの属性としてrdma利用属性を指定する

```
[node01]# gluster volume create vol1 transport rdma
node01:/export/brick1 node02:/export/brick1
```

- rdmaとtcpの両方を指定することも可能

```
[node01]# gluster volume create vol1 transport rdma,tcp
node01:/export/brick1 node02:/export/brick1
```

ボリューム属性変更(1)

- クォータ設定
  - ディレクトリ単位でクォータを設定できる
  - 論理ボリューム内での絶対パスを指定して設定

```
[node01]# gluster volume quota vol4 limit-usage / 50GB
limit set on /

[node01]# gluster volume quota vol4 limit-usage /foo 30GB
limit set on /foo
```

ボリューム属性変更(2)

- クォータ設定確認

```
[node01]# gluster volume quota vol4 list
        path                  limit_set          size
-------------------------------------------------------------------------------
/                             50GB               20.6GB
/foo                          30GB               18.2GB
```

ボリューム属性変更 – データ保全（1）

- 遅延書き込み(write behind)を禁止する
  - 標準設定では、アプリケーションのwriteシステムコール完了（成功）後に、バックグラウンドでGlusterサーバ上のBrickへの書き込みを開始する
  - I/O性能とのトレードオフ

```
[node01]# gluster volume set vol4 performance.flush-behind off
Set volume successful
```

ボリューム属性変更 – データ保全（2）

- Split Brain対策
  - 冗長ボリュームを構成するBrickの一定数以上への書き込みが行えないと、更新処理（writeシステムコールなど）は成功しない
  - GlusterFS 3.3 以降で利用可能

例）半数以上のBrick更新を保証

```
[node01]# gluster volume set vol4 cluster.quorum-type auto
Set volume successful
```

ボリューム属性変更 – データ保全（3）

- Split Brain対策

例）指定数以上のBrick更新を保証

```
[node01]# gluster volume set vol4 cluster.quorum-type fixed
Set volume successful

[node01]# gluster volume set vol4 cluster.quorum-count 2
Set volume successful
```

ボリューム属性変更 – タイムアウト時間

- サーバダウン検出時間変更
  - 標準設定では 42秒以上応答がないとサーバダウンとみなす

例）タイムアウトを22秒にする

```
[node01]# gluster volume set vol4 network.ping-timeout 22
Set volume successful
```

ボリューム属性変更 – キャッシュサイズ

- キャッシュサイズを、標準設定の32MBから128MBに拡大する

```
[node01]# gluster volume set vol4 performance.cache-size 128MB
Set volume successful
```

- 100MB以上のサイズのファイルはキャッシュしない（標準設定では全てキャッシュする）

```
[node01]# gluster volume set vol4 performance.cache-max-file-size 100MB
Set volume successful
```

情報取得 – 論理ボリューム

```
[node01]# gluster volume status vol4 detail
Status of volume: vol4
------------------------------------------------------------------------------
Brick                : Brick node01:/export/brick5
Port                 : 24011
Online               : Y
Pid                  : 1254
File System          : ext4
Device               : /dev/vdb
Mount Options        : rw,user_xattr
Inode Size           : 256
Disk Space Free      : 92.4GB
Total Disk Space     : 92.9GB
Inode Count          : 6193152
Free Inodes          : 6130970
------------------------------------------------------------------------------
Brick                : Brick node02:/export/brick5
Port                 : 24011
Online               : Y
                       (略)
```

情報取得 – アクセス統計（1）

- プロファイリング
  - ブリック単位でアクセス数、アクセス種別の情報を採取できる

プロファイリング開始

```
[node01]# gluster volume profile vol4 start
Starting volume profile on vol4 has been successful
```

情報取得 – アクセス統計（2）

- プロファイリング結果出力

```
[node01]# gluster volume profile vol4 info
Brick: node01:/export/brick5
--------------------------------
Cumulative Stats:
     Block Size:               4b+
 No. of Reads:                 1
No. of Writes:                14
 %-latency   Avg-latency   Min-Latency   Max-Latency   No. of calls       Fop
 -----------   -----------   -----------   -----------   ------------      ----
      0.01      78.00 us      78.00 us      78.00 us             61     READ
      0.02      49.75 us      26.00 us      59.00 us              4    FSTAT
      0.03      36.14 us      19.00 us      48.00 us             27     STAT
      0.04      24.71 us      19.00 us      50.00 us             24   ACCESS
      0.04      62.67 us      57.00 us      68.00 us             96    WRITE
                     :
     Duration: 67884 seconds
    Data Read: 4 bytes
Data Written: 56 bytes
                     :
```

情報取得 – アクセス統計（3）

- ファイル単位のアクセス状況表示
  - アクセスタイプ(read, writeなど)を指定できる

```
[node01]# gluster volume top vol4 write
Brick: node01:/export/brick5
Count                filename
=======================
3                    <gfid:0f7a55c4-3988-4f2a-905d-4392ac510cd7>/file9705
2                    <gfid:0f7a55c4-3988-4f2a-905d-4392ac510cd7>/file9706
1                    /foo/add9763
1                    /foo/add9762
          :
Brick: node03:/export/brick5
Count                filename
=======================
3                    /foo/file9705
2     /foo/file9706
          :
```

情報取得 – 動作ログ

- 各 Glusterサーバの /var/log/glusterfs 配下に、glusterfsの動作ログが出力される
- 各Brickに毎の操作ログは、/var/log/glusterfs/bricks 配下に出力される

遠隔レプリケーション（１）

- 異なるロケーションにあるGlusterFSシステムに論理ボリュームの複製(スレーブ）を置くことができる
  - 複数のスレーブを指定可能
  - カスケード接続も可能
    - スレーブの下に更にスレーブをぶら下げることができる
  - スレーブとしてローカルディスクも指定可能
  - 自動同期設定も可能（デフォルトは手動起動）
- バックアップ、DR用途
- Gluster 3.2以降で利用可能

遠隔レプリケーション（2）

- Glusterサーバに下記パッケージを追加インストール
  - 遠隔レプリケーション用glusterfsパッケージ
  - ssh、rsync、pythonパッケージ
    rsync over sshで、レプリケーション処理を実現しているため

```
# yum install --nogpgcheck glusterfs-geo-replication-3.3.1-1.el6.x86_64.rpm

# yum install python openssh-clients rsync
```

- マスター（バックアップ元）のGlusterサーバ群の時刻は、正確に揃えておく
  - 推奨：NTPの利用

遠隔レプリケーション（3）

- レプリケーション処理用ユーザの作成
  - スレーブとして動作するGlusterサーバ上に専用のユーザアカウントを作成
    - uid は 0 ― root権限を与える
    - ユーザ名は任意 （ここの例では geosync とする）

```
[slave03]# useradd -o -u 0 geosync
```

遠隔レプリケーション（４）

- ssh鍵の交換
  - 遠隔レプリケーション処理を起動するマスター側のGlusterサーバでsshを生成する
    - パスワードなしのssh鍵を作成

```
[node01]# ssh-keygen -P "" -f /var/lib/glusterd/geo-replication/secret.pem
```

  - sshの公開鍵を、スレーブ側のGlusterサーバに配布する

```
[node01]# ssh-copy-id -i /var/lib/glusterd/geo-replication/secret.pem.pub
geosync@slave03
```

遠隔レプリケーション（５）

- アクセス制御
  - スレーブGlusterサーバにて、遠隔レプリケーション用のアカウントgeosyncに対するsshログインからは、遠隔レプリケーション処理以外は行えないようにする
    - ssh鍵ファイルに、実行するコマンドとして、/usr/libexec/glusterfs/gsyncd を登録する

```
[slave03]# vi ~geosync/.ssh/authorized_keys

command="/usr/libexec/glusterfs/gsyncd“ ssh-rsa AAAAB3NzaC1.....
```

遠隔レプリケーション（６）

- スレーブGlusterサーバにて、レプリケーション処理時にスレーブボリュームをマウントする作業ディレクトリを用意
  - ディレクトリ名は任意
    ここでは、/var/mountbroker-root とする
  - アクセス権は、0711

```
[slave03]# mkdir /var/mountbroker-root

[slave03]# chmod 0711 /var/mountbroker-root
```

遠隔レプリケーション

- 実際の遠隔レプリケーション処理を実行するグループとユーザをスレーブGlusterサーバ上に作成する
  - グループ名は任意 （ここでは仮に geogroup とする）
  - ユーザ名は任意
    - 一般ユーザ、特権は不要
    - このスレーブサーバにバックアップを行うマスターが複数ある場合、マスターごとに別のユーザを定義するとよい（推奨）

```
[slave03]# groupadd geogroup

[slave03]# useradd -g geogroup geoaccount0

[slave03]# useradd -g geogroup geoaccount1
```

遠隔レプリケーション

- スレーブボリューム（バックアップ先）を、必要な数だけ予め用意しておく
  - ストレージタイプは任意
    バックアップ元のデータの重要度に合わせて決める

```
[slave03]# gluster volume create vol0_slave slave01:/export/brick0 slave02:/export/brick0

[slave03]# gluster volume create vol1_slave slave01:/export/brick1 slave02:/export/brick1
        ……
```

遠隔レプリケーション

- スレーブボリュームを登録する
  - スレーブGlusterサーバの/etc/glusterfs/glusterd.volファイルに、スレーブボリュームを登録する
    - 操作を行うユーザと対にして登録

```
volume management
    type mgmt/glusterd
    option working-directory /var/lib/glusterd
    option transport-type socket,rdma
    option transport.socket.keepalive-time 10
    option transport.socket.keepalive-interval 2
    option transport.socket.read-fail-log off

    option mountbroker-root /var/mountbroker-root
    option mountbroker-geo-replication.geoaccount0 vol0_slave,vol1_slave
    option mountbroker-geo-replication.geoaccount1 storeX_backup
    option mountbroker-geo-replication.geoaccount2 diskX_slave,diskY_slave
    option geo-replication-log-group geogroup
end-volume
```

遠隔レプリケーション

- マスターGlusterサーバにて、レプリケーション開始を指示

```
[node01]# gluster volume geo-replication vol0 geosync@slave03::vol0_slave start
Starting geo-replication session between vol0 & geosync@slave03::vol0_slave has been successful
```

遠隔レプリケーション

- レプリケーション状態の監視

```
[node01]# gluster volume geo-replication vol0 geosync@slave03::vol0_slave status
MASTER          SLAVE                                   STATUS
-------------------------------------------------------------------------------
vol0            geosync@slave03::vol0_slave             starting...
```

  - レプリケーション完了

```
[node01]# gluster volume geo-replication vol0 geosync@slave03::vol0_slave status
MASTER          SLAVE                                   STATUS
-------------------------------------------------------------------------------
vol0            geosync@slave03::vol0_slave             OK
```

遠隔レプリケーション

- マスターGlusterサーバにて、レプリケーション終了を指示

```
[node01]# gluster volume geo-replication vol0 geosync@slave03::vol0_slave stop
Stopping geo-replication session between vol0 & geosync@slave03::vol0_slave has been successful
```

開発動向

- GlusterFS 3.4に向けて様々な提案が行われている
  - VM環境との親和性向上
    - qemu連携機能
    - ブロックデバイスインターフェイス
  - Split Brain対策機能強化
  - 異なるバージョンのGlusterFSサーバの混在
  - Write Onceボリューム
  - マネジメントツール(oVirtやpuppetなど)との連携
  - IDAアルゴリズム導入:ミラーボリュームでなく、RAID5やRAID6の論理ボリュームを可能とする
  - 性能改善、効率化、安定性の向上

  など……