VCPを利用したAWS上でのMoodle環境の構築・運用

浜元信州

群馬大学総合情報メディアセンター

横山重俊[1),2)]，竹房あつ子[2)]，合田憲人[2)]

1)群馬大学総合情報メディアセンター

2)国立情報学研究所

Moodleについて

- 日本の大学32.7%で利用（学部研究科）
  - 利用率は他のLMSと比較してトップ
  - AXIES：高等教育機関等におけるICT利活用に関する調査研究（H28）
- オープンソース：無料
  - Linux，Apache，MySQL，PHP（LAMP環境）で動作
- 自習等の目的で学外公開されていることが多い

Moodle運用上の課題

コース作成

コース登録方法
（一括登録/コース申請）

授業以外での利用

プラグイン

アップデートの妨げ

インストール基準

高負荷対応

チューニング
（http, SQL, PHP）

リソース増強
（CPU，メモリ，HDD）

IaaSクラウドの利用が有効?

質問

教務系の質問
（履修生，担当教員管理）

Moodleテクニカルな質問
（モジュール利用方法）

サーバ基盤系の質問

脆弱性

Moodleアップデート

ミドルウエア
（LAMP環境）
アップデート

ハードウエア

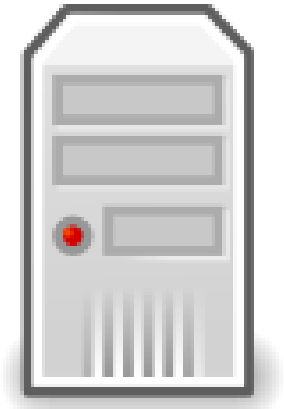
障害対応

寿命対応

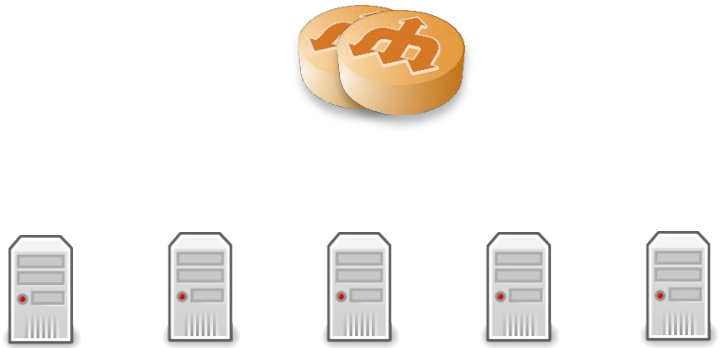
IaaS型クラウド

- ハードウエア保守
  - オンプレミス＋オンサイト保守とほぼ同じ
  - ハードウエア寿命対策でできることは含まれている

- ハードウエア資源の確保が容易



高スペックサーバ



冗長構成

Virtual Cloud Provider

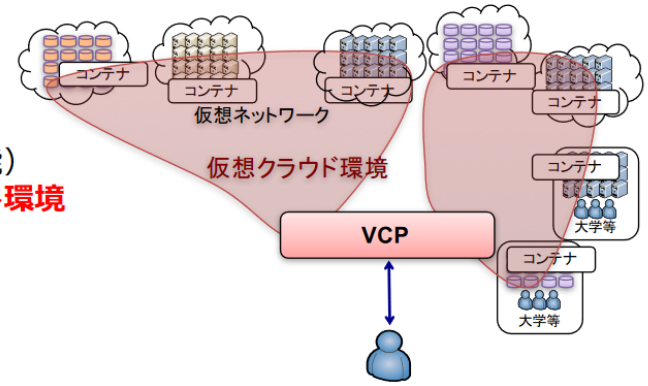
クラウドによらないAPI操作をNotebook経由でも提供

NII 竹房先生@オープンフォーラム2017

インタークラウドサービス

- NII開発の**Virtual Cloud Provider（VCP）**ソフトウェアにより、SINET接続の仮想クラウド環境を提供
  - SINET接続拠点間ネットワークの接続設定の技術的支援
  - Dockerコンテナを活用した仮想クラウド環境の構築
  - クラウドプロバイダごとのインタフェースの差異を吸収するAPI提供

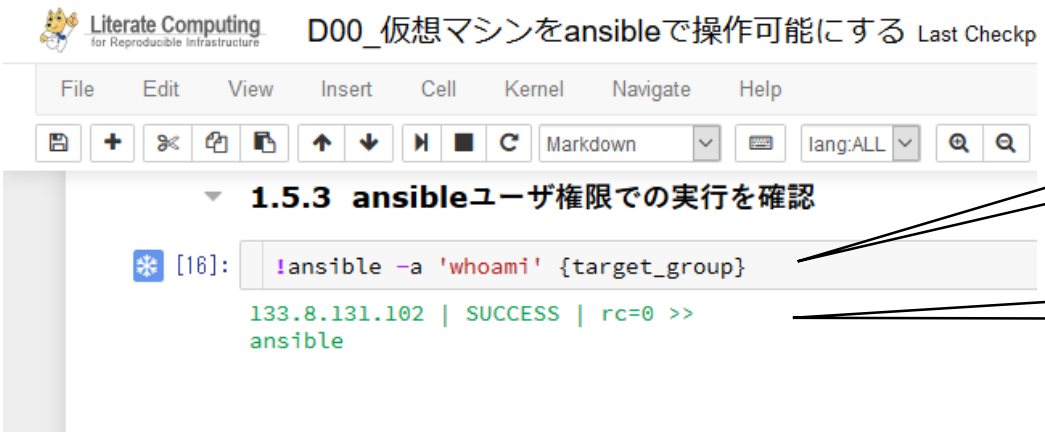
ユーザからの要求
（拠点、資源量、ネットワーク性能）
に応じて**ユーザ専用の仮想クラウド環境**
をオンデマンドに作成



National Institute of Informatics

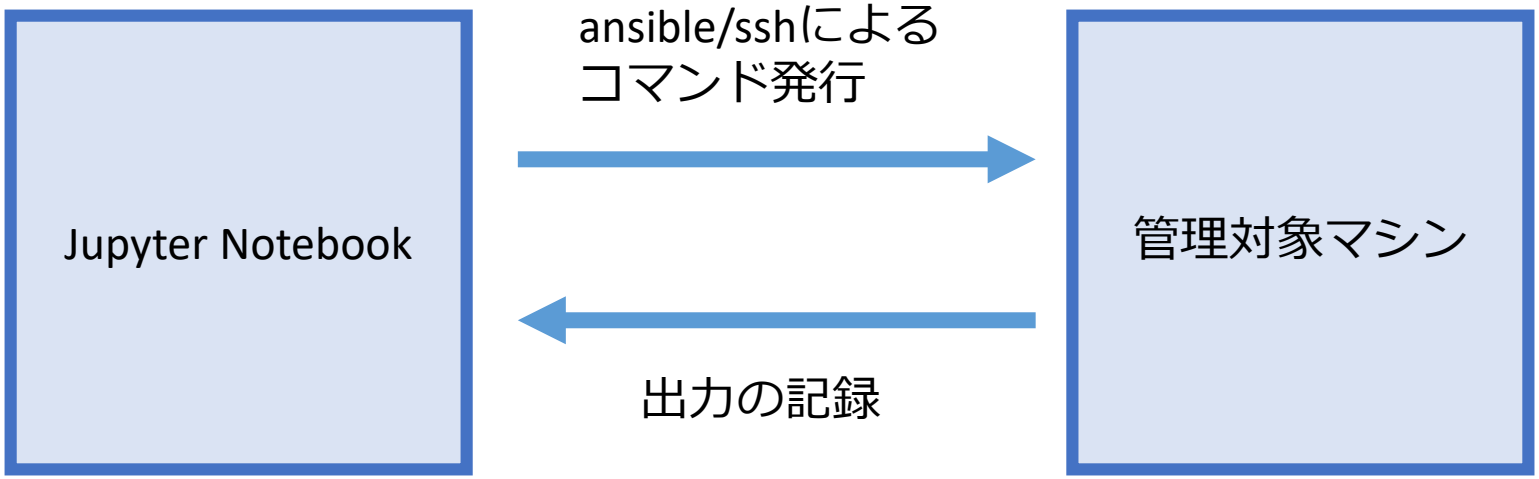
Jupyter Notebookによる運用

作業手順／設定をNotebookによって共有できる



手順・コマンド

結果



Moodle運用上の課題

コース作成

コース登録方法
（一括登録/コース申請）

授業以外での利用

プラグイン

アップデートの妨げ

インストール基準

Notebook

高負荷対応

チューニング
（http, SQL, PHP）

リソース増強
（CPU，メモリ，HDD）

Notebook

IaaS

IaaSクラウドの利用が有効

質問

教務系の質問
（履修生，担当教員管理）

Moodleテクニカルな質問
（モジュール利用方法）

サーバ基盤系の質問

脆弱性

Moodleアップデート

ミドルウエア（LAMP環境）アップデート

Notebook

ハードウエア

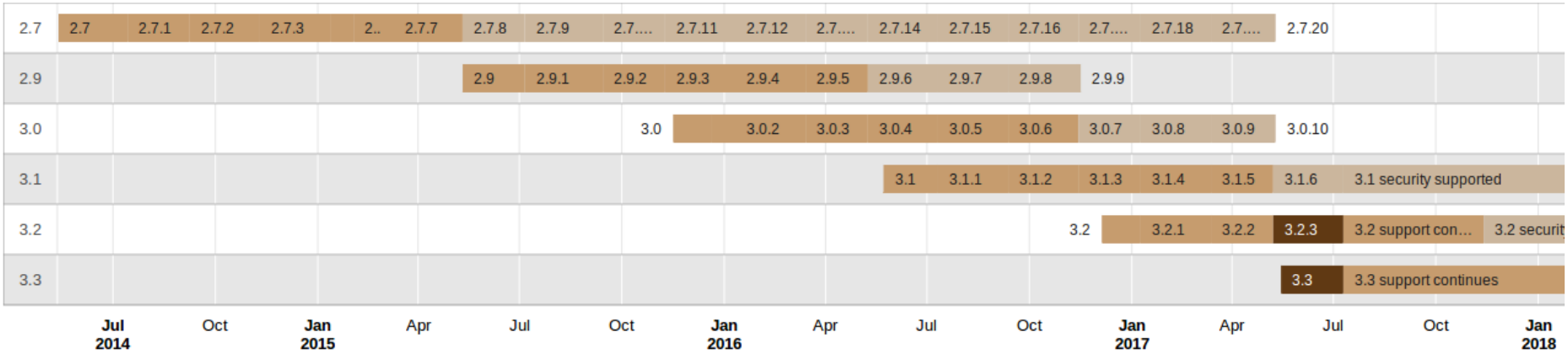
障害対応

寿命対応

IaaS

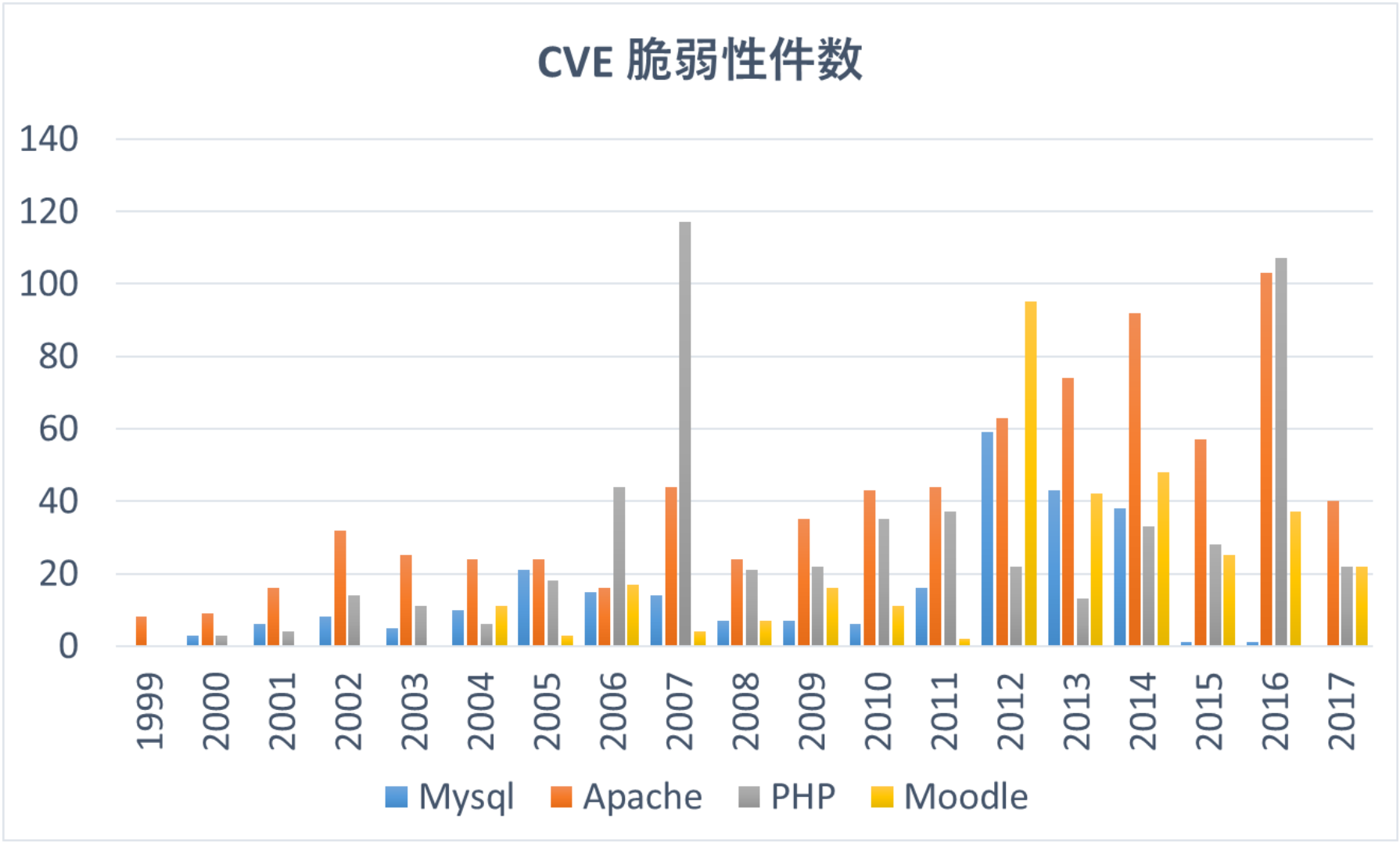
Moodleリリーススケジュール

- Moodle2.6以降リリーススケジュールが固定化
- マイナーバージョンは6カ月ごとにリリース（5月と11月第2週月曜）
- 3カ月に一度のペースでアップデート
- サポート期間は通常18カ月，LTSは36カ月
  - BugFix 12カ月＋Security Update 6カ月or24カ月



http://www.cvedetails.com/

脆弱性件数の推移



アップデートの要求要件

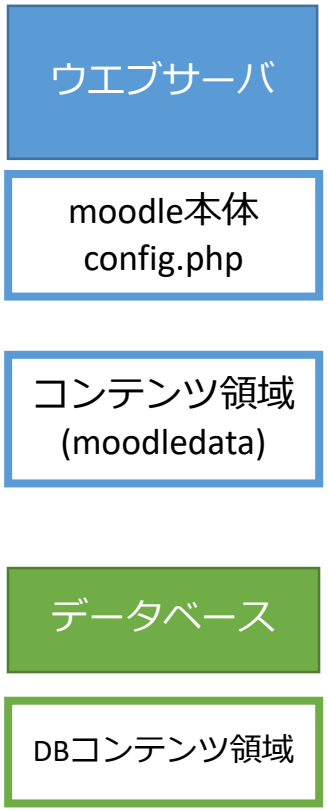
1. （理想的には）サービスを止めない。
   - できる限りサービスは止めたくない
2. 万が一，失敗しても回復させたい。
3. アップデートの手順は手間をかけたくない。
4. アップデートは複雑でなく，分かりやすく。

目標

IaaS型パブリック/プライベートクラウドを利用して

1. Moodleの構築・アップデートを（わかりやすい範囲で）自動化する
2. Moodleのアップデートでは，検証・切戻できるようにする
    1. 事前検証環境作成
    2. 検証環境のバージョンアップ
    3. 本番環境のバージョンアップ
    4. アップデート失敗時の切り戻し
3. Moodleアップデートに伴う停止を短くする

Moodleのアップデート手順

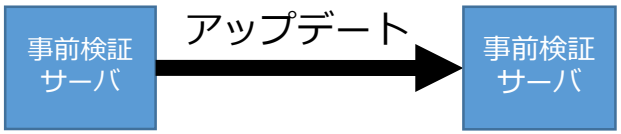


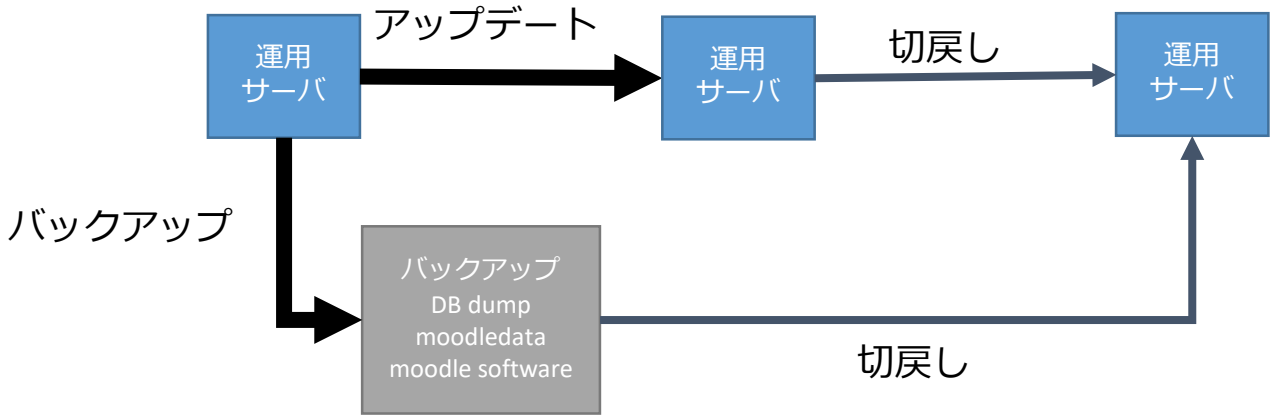
1. メンテナンスモードに切り替え
   スクリプトが準備されている（maintainance.php）
2. Moodle本体をアップデート
   gitによる自動化
3. データベースのアップデート
   ウエブアクセスの必要があるが，自動化されている
   (upgrade.php)
4. プラグインのアップデート
   ウエブアクセスの必要があるが，自動化されている
5. 新パラメータの調整
   ウエブ経由
6. メンテナンスモードから回復
   スクリプトが準備されている（maintainance.php）

- かなりの部分が自動化されている。
- アップデート後の新パラメータの設定は手動。
- メンテナンスモードに入るためログアウト必要。
  →無停止アップデートは難しい
- 失敗時の切り戻しは考慮されていない。

切戻しの方法（バックアップ利用）

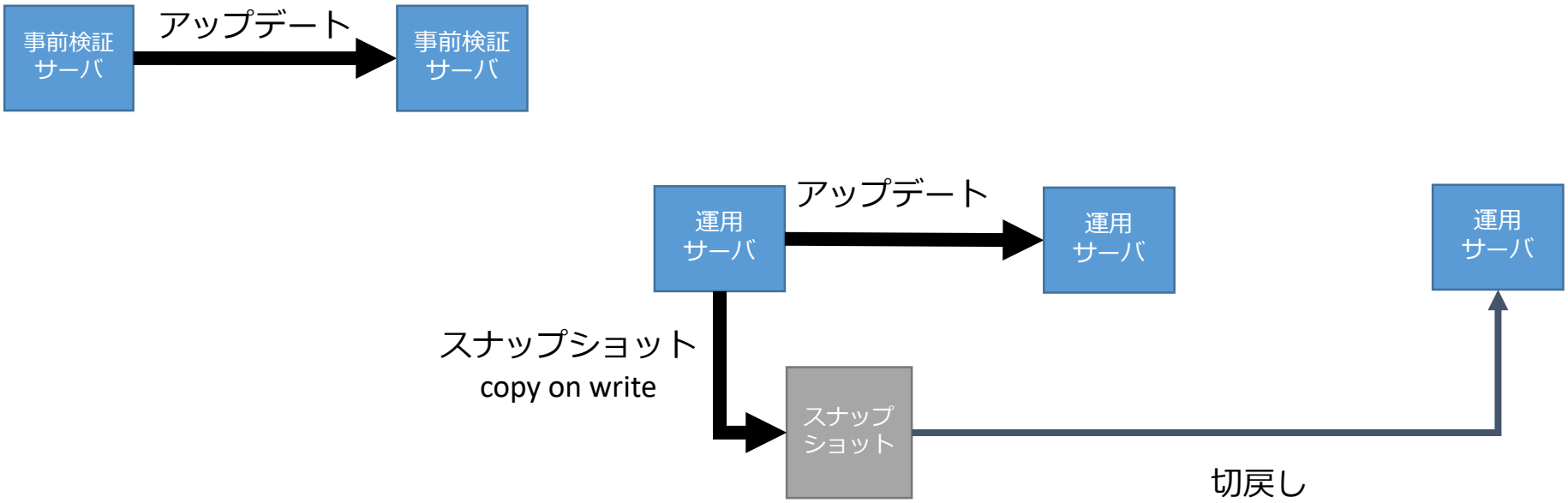
- バックアップは取得しておく：
  DB，moodle script，moodledataは，運用サーバとは物理的，地理的に異なる場所にアーカイブしておく。
- アップデートした環境を捨てて，アーカイブデータを利用して，再構築する。





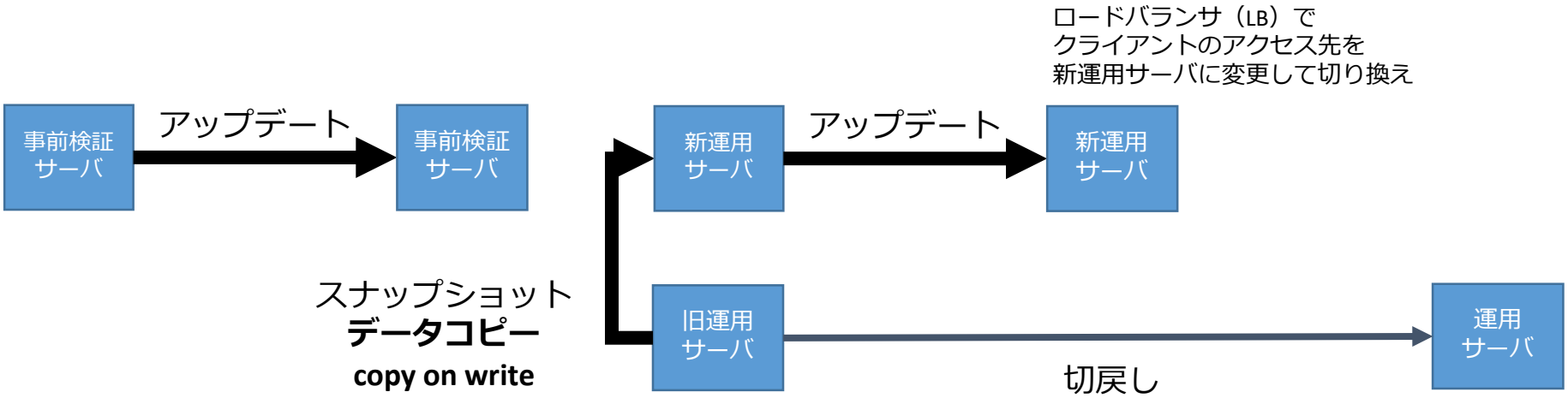
切戻しの方法（スナップショット利用）

- ファイルシステムのスナップショット機能を利用する。サーバを止めてスナップショットを取得し，サーバを起動して，問題が起こったらスナップショットに戻す。



切戻しの方法（Blue-Green deployment）

- サーバを2台準備し，データのコピーを作成してから，1台をアップグレードする。問題が起こったら，元のサーバに戻す。



方法の比較

方法	復旧点 取得方法	復旧点 取得時間	切戻し 時間	サーバ資源
Backup	コピー	長	長	3台 運用・検証・ バックアップ
Snapshot	Copy on write	短	中	3台 運用・検証・ バックアップ
Blue-Green	Copy on write	短	短	4台 運用・検証・LB・ バックアップ

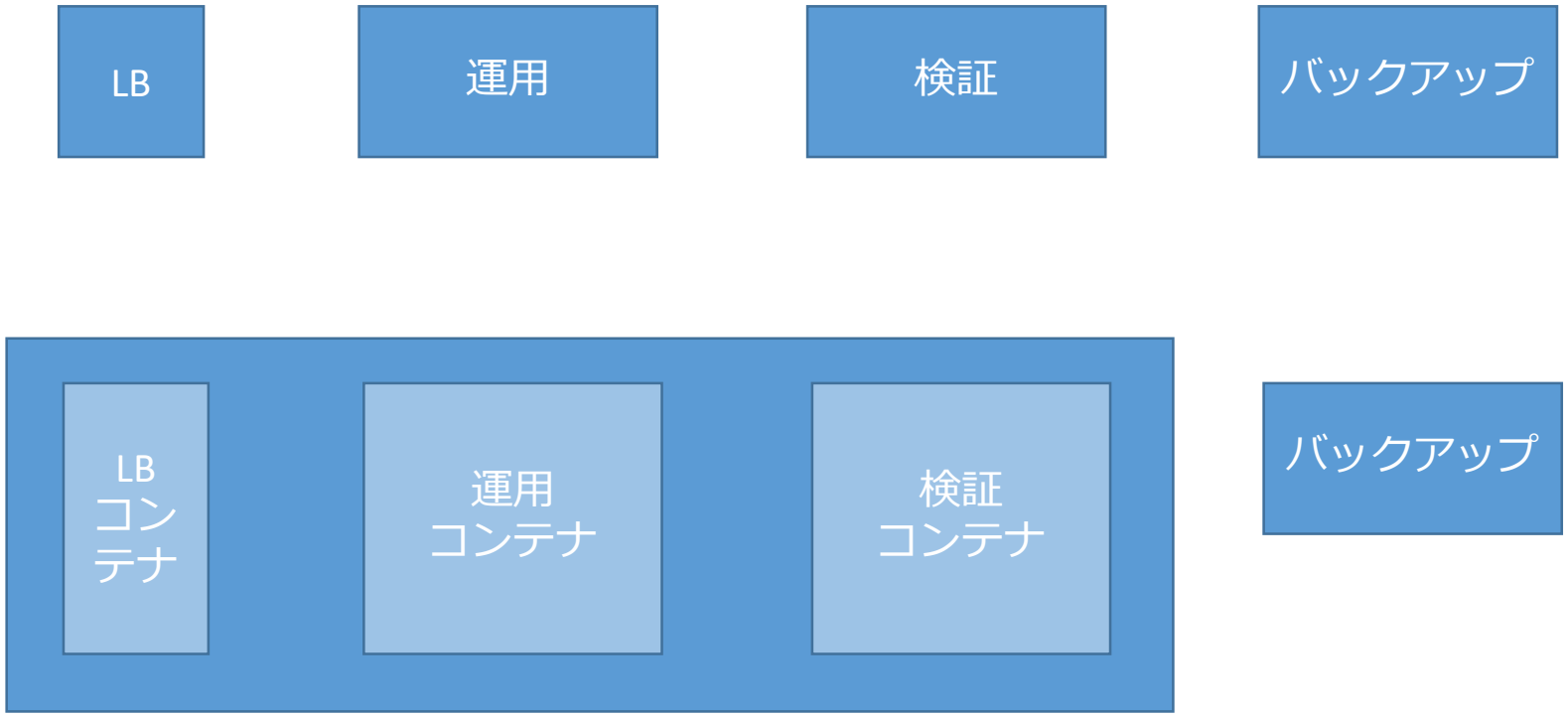
Dockerの利用

- コンテナ型のアプリケーション実行環境

- 各コンテナの起動が早い
  - ホストから見ると１つのプロセス
  - アップデート時間の短縮につながる

- コンテナー一つ一つのリソースは仮想マシンに比べると少ない
  - 運用，検証環境等をまとめることができる。

Dockerの利用

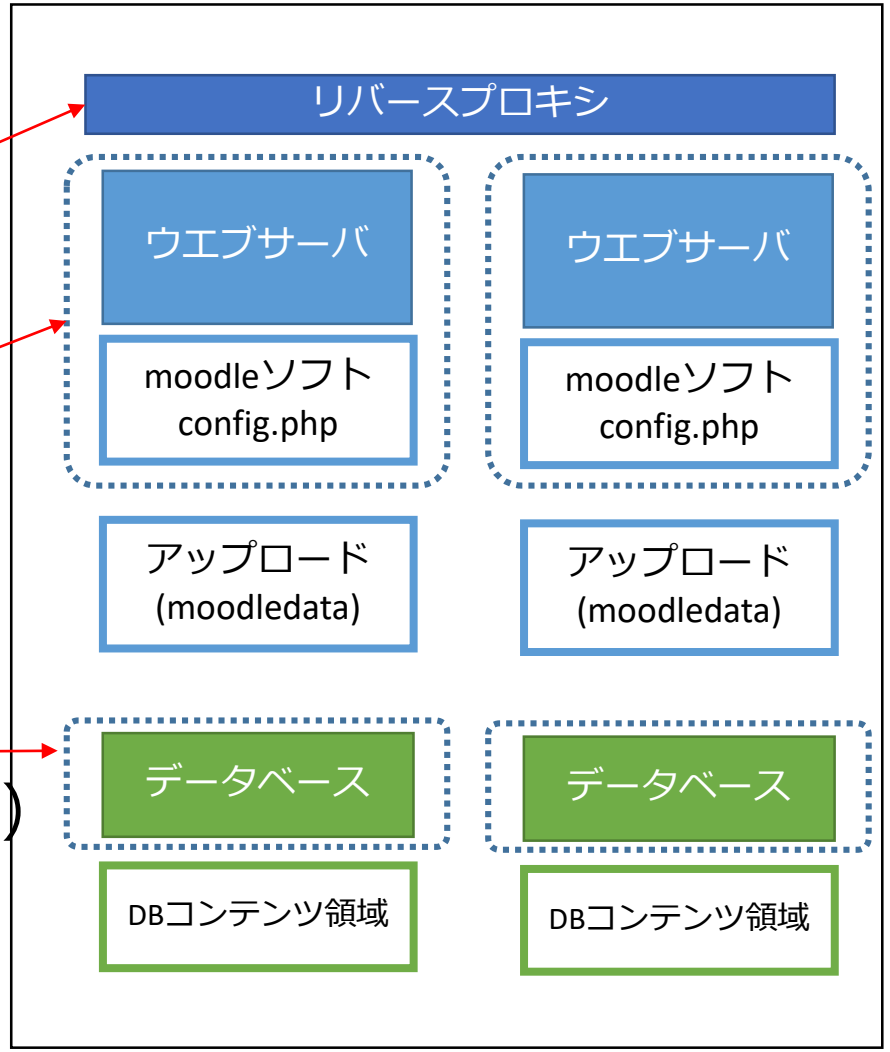


方法の比較

方法	復旧点 取得方法	復旧点 取得時間	切戻し 時間	サーバ資源
Backup	コピー	長	長	3台 運用・検証・ バックアップ
Snapshot	Copy on write	短	中	3台 運用・検証・ バックアップ
Blue-Green	Copy on write	短	短	4台 運用・検証・LB・ バックアップ
Blue-Green on Docker	Copy on write	短	短	2台 運用検証LB・ バックアップ

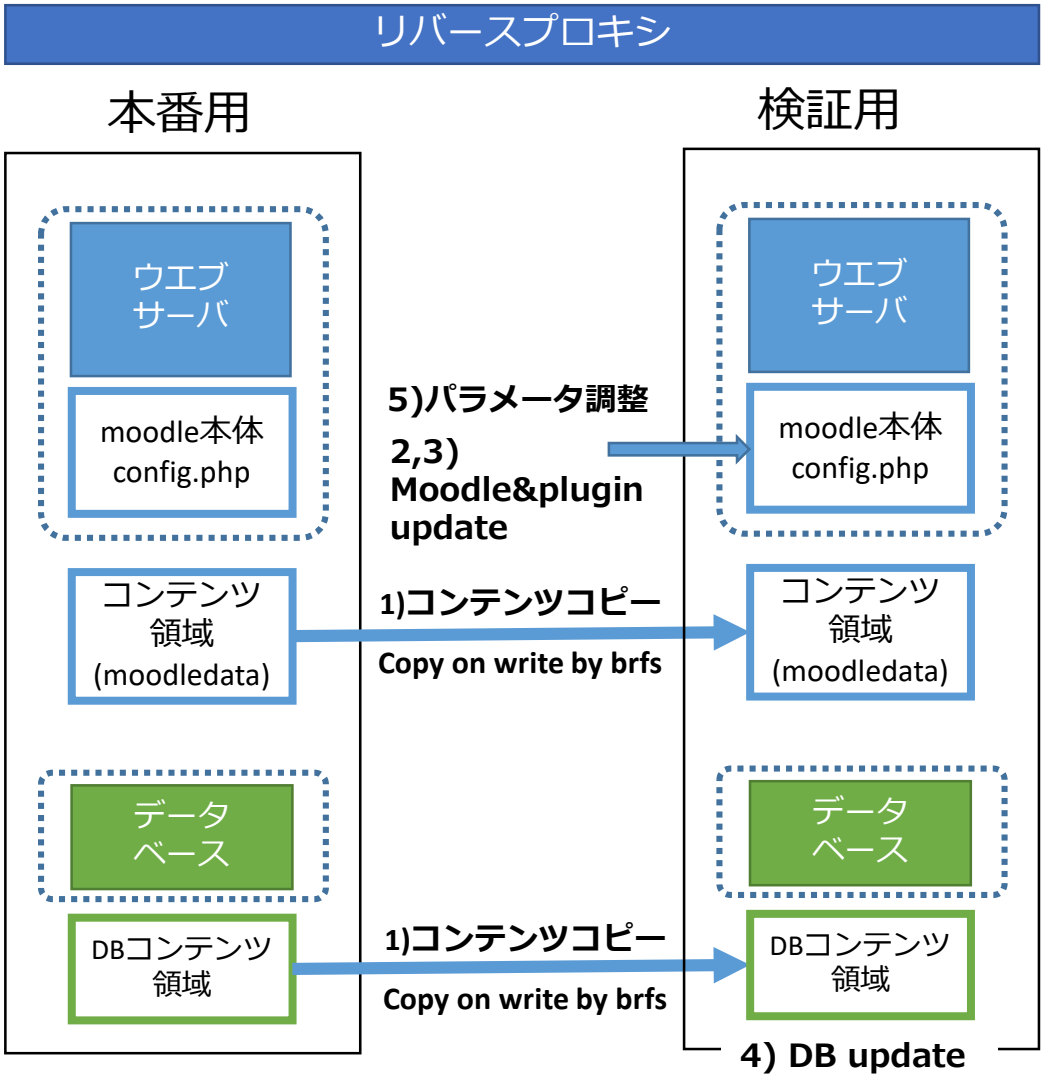
アップデートの詳細

- リバースプロキシコンテナ
  - apache
  - （shibboleth）
- Moodleコンテナ
  - CentOS7/php,apache
- データベースコンテナ
  - mysqlコンテナ（@DockerHub）



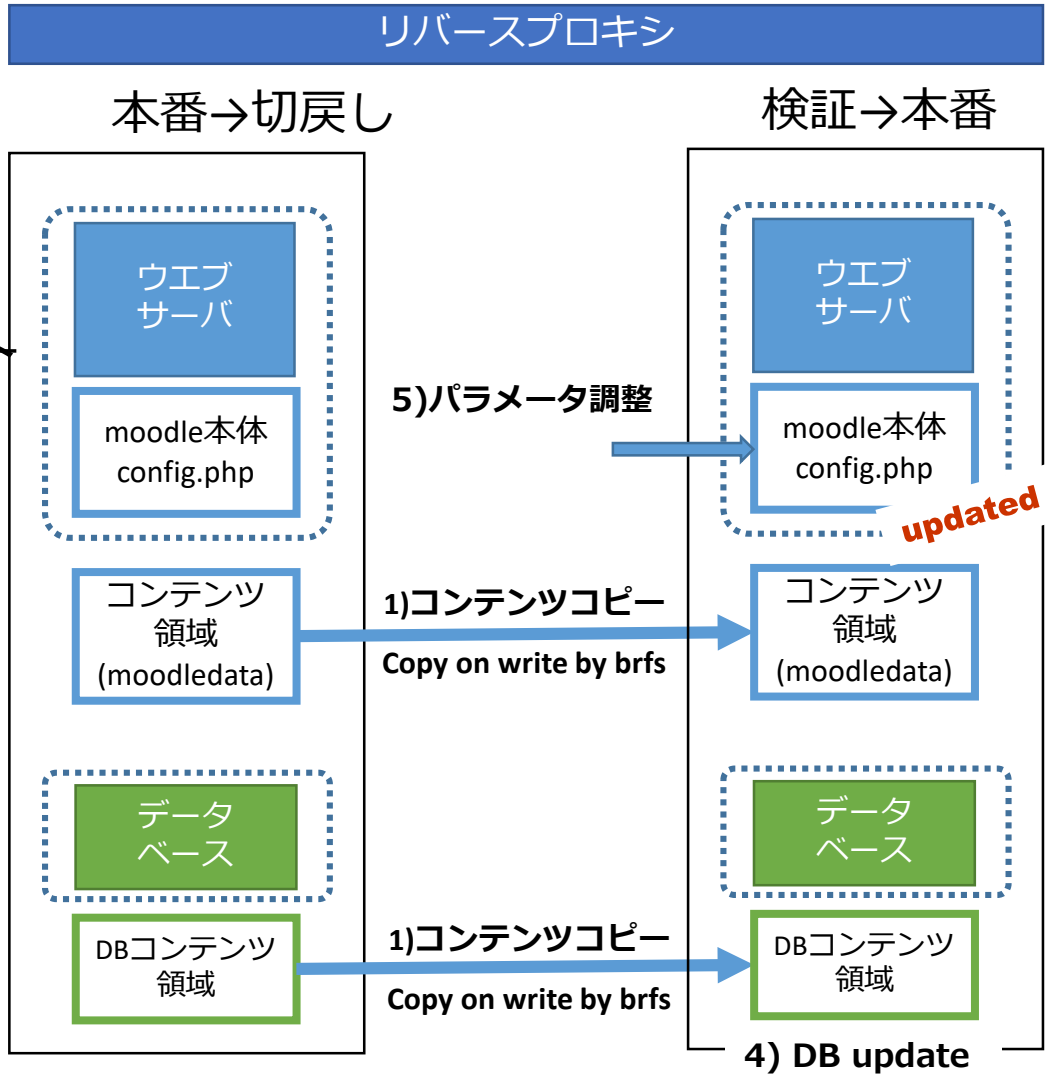
検証用サーバをつくる

1. コンテンツコピー
   →**copy on writeによる高速化**
2. moodleアップデート
3. プラグインアップデート
4. DBアップデート
5. パラメータ調整



検証から本番への移行

1. コンテンツコピー
2. ~~moodle/LAMPアップデート~~
3. ~~プラグインアップデート~~
4. DBアップデート
5. パラメータ調整
6. リバースプロキシ切替



検証結果１：検証環境作成

- AWS
  - EC2インスタンス　m4.large
  - Filesystem btrfs

操作内容	要した時間	容量
DBコンテンツ領域の削除	4s	254MB
DBコンテンツ領域のコピー	1s	
アップロードファイル領域の削除	4s	1.1GB
アップロードファイル領域のコピー	1s	
Moodleソフトウエア領域の削除	3s	560MB
Moodleソフトウエア領域のコピー	2s	

検証結果２：アップデート

操作内容	要した時間
メンテナンスモード切替	4s
コンテンツ領域のコピー	1s
アップロードファイル領域のコピー	2s
新環境コンテナの起動・確認	5s
メンテナンスモード解除	1s
DBのアップデート	58s
LBの設定変更・確認	5s

実演とまとめ

- 小規模構成を想定して，Jupyter Notebookを利用したMoodleのアップデート手順を作成した。
- ダウンタイムが極力短くなるよう，copy on writeによる高速化，Dockerによるイメージ化を行った。
- Dockerコンテナの利用により，現実に近い事前検証環境を同一サーバ内準備することができた。
- アップデートと切り戻しを確実に行うため，リバースプロキシによる切り替えを行う構成とした。

謝辞

- 本発表で使用したクラウド資源は，平成29年度国立情報学研究所「クラウド利活用実証実験」から提供を受けています。