Vol.2016-IOT-35 No.13
Vol.2016-SPT-20 No.13
2016/9/24

Mutable データ移行による計算機環境再構築

石坂徹[†1]　桑田喜隆[†1]　横山重俊[†2]　合田憲人[†2]

概要：
論文等の検証環境の提供や情報システムの移行など，計算機環境そのものを再現する需要がある．仮想化環境の発達により，計算機環境そのものを提供することが可能になった．最も単純な方法は仮想基盤から直接イメージを取り出し，それを提供することである．しかし，再現時にソフトウェアのアップデートなどが必要な場合も多く入手したイメージそのものを，そのまま実行できないことも考えられる．また，インターネットリソース，ストレージ容量は無限ではなく，提供するデータは出来るだけ少なくする必要がある．本研究では，コンテナ技術を利用した仮想化基盤を用いて，計算機環境を提供し，再現するための手法を提案する．さらに Web アプリケーションのような複数の計算機環境からなるシステムへの応用を行い，実証を行った．

キーワード：システム再構築，仮想化基盤，コンテナ

Reconstruction of Computing Environment by Migration of Mutable Data

Tohru Ishizaka[†1], Yoshitaka Kuwata[†1], Shigetoshi Yokoyama[†2], Kento Aida[†3]

Abstract:
There is a need to reconstruct the computing environment, such as a verification of experiment and migration of information system. By the development of the virtual environment, it has become possible to provide a computing environment itself. The simplest way in order to provide computing environment is that retrieves images directly from a virtual infrastructure. However, it is also often necessary, such as software updates at the time of reconstruction, it is also conceivable that cannot be run as it is. Also, Internet resources, the storage capacity rather than infinite, data provided, it is necessary to reduce as much as possible. In this study, we propose the efficient method for provide a computing environment and reconstruct it, using the using container technology. Further we are applied this method to a system including a multiple machines such as Web applications.

Keywords: system reconstruction, virtualization infrastructure, container

1. はじめに

仮想化基盤の利用が研究室や個人など小規模な組織でも利用できるようになり，計算機環境そのものをやり取りすることが可能になっている．計算機環境をやり取りする場面のひとつとして，研究における実験結果の再現がある．実験結果を再現することは，その実験が正しいことを保証するもっとも明快な手法である．多くの場合，研究環境の成果の公表は論文，特許など限られた紙面内で行われる．従って，その研究内容の詳細は公表物とは別にインターネットのウェブサイトやリポジトリ，あるいは個人的なやりとりなどで提供される．横山ら[1]は，論文再現環境として，複数のクラウド基盤にまたがった Overlay 環境を構築している．計算機を用いた研究においては，データだけでなく，計算に用いたライブラリ，ミドルウェアなどの環境が提供されないと正しく実験の再現ができないことも考えられる．また，仮想マシン上で実験を行い，その仮想マシンイメージをそのまま提供することも可能であるが，提供先でその仮想マシンイメージをそのまま実行できないケースも考えられる．例えば，仮想マシンイメージで用いられている OS やライブラリにセキュリティ上の問題があり，提供先での動作が禁止される場合，あるいはライブラリ自体に不具合があることが判明しており，それらをアップデートする必要がある場合などである．

また，もう一つのユースケースとして，情報システムサービスの移行が挙げられる．情報サービスでは Web 三層モデルのようにユーザインターフェース，アプリケーション，データベースのように機能ごとにマシンを分離することが多い．さらにこれらが可用性のため冗長化されている場合もあり，OS やミドルウェア，サービスプログラムのアップデートには多大な作業量が必要となる．

そこで，本研究ではシステムの再構築時に OS やライブラリを最新の状態で利用する必要があることを想定して，情報システムの再現に必要な環境を保存し，再構築する手法を提案する．

†1 室蘭工業大学
Muroran Institute of Technology.
†2 群馬大学
Gunma University
†3 国立情報学研究所
National Institute of Infomatics

Vol.2016-IOT-35 No.13
Vol.2016-SPT-20 No.13
2016/9/24

2. 情報システムの提供方法

従前用いられている計算環境の提供は様々な方法が用いられている. 以下に主に用いられている提供方法を述べる.

2.1 ソースコード及びデータの提供

実験を行ったプログラムのソースコードを提供することは最も手軽な方法である. Unix 系システムであれば, ソースコードとデータをまとめて圧縮した tarball 形式がよく使われる. この場合, 被提供者は tarball を展開し, 必要に応じてソースコードのコンパイルなどを行うことで再現が可能になる. この作業が複雑な場合は, 環境の提供者はあらかじめ Makefile などの構築を自動的に行うための構築手順を用意する必要がある. さらに必要なライブラリの情報などは別途, 利用環境に関する文書を作成して提供する必要がある.

2.2 仮想マシンイメージの提供

多くの仮想化基盤はイメージ提供のためのツールを備えており, 提供者は簡易に提供するイメージを作成することができ, また, 被提供者も同様に再生することができる. これにより, 被提供者は環境を完全に再現することができるが, 環境に関する情報を取得するには被提供者が OS やライブラリ, ミドルウェアに関する知識が必要である. 前述のようにイメージをそのまま再生できない場合は, これらのソフトウェアのバージョンアップ, 設定などの作業が発生し, 被提供者の負担は大きくなる.

また, イメージは OS だけでも数百 MB 以上となり保存のための領域が大きくなる.

図 1　一般的なシステム再構築手法（ディスクイメージ提供の場合）

Figure 1　Common Method of System Reconstruction (Case of Disc-Image Provided)

3. 本手法の概要

3.1 Mutable データの分離

本稿では, 前節で述べた環境提供手法の欠点を解消するため, 環境構築手順を提供する手法を提案する. この手法では, OS やアプリケーションプログラムなど入手することが比較的容易なものは, 構築時に外部から取得し, 提供するデータ等を出来るだけ少なくする. 例えば, Web アプリケーションプログラムの研究・開発を行った環境を提供する場合は, Web アプリケーションそのもののデータはそのまま提供し, OS や Web サーバプログラム, ミドルウェア等については, ソフトウェアの情報とシステムの構築手順を構成情報として提供する. 図 2 に概念図を示す.

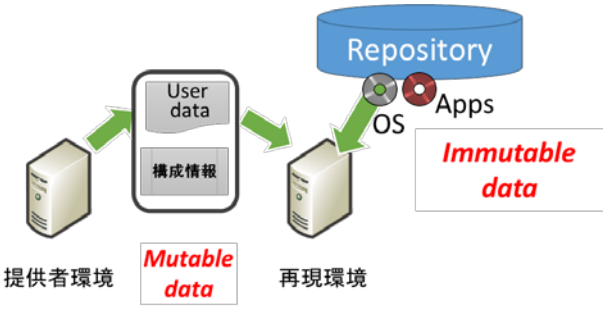

図 2　提案手法

Figure 2　Proposed Method

本手法で再現のために提供されるデータを Mutable データと呼ぶ. これは運用開始から提供するまでの間に, 生成, 改変されたすべてのデータである. それに対して, 外部のリポジトリ等から取得する OS, アプリケーション等は Immutable データと呼ぶ. これら Mutable/ Immutable データの分離のための基準を表 1 に基準を示す.

表 1　Mutable データ分離の基準

Table 1　Criteria for Separation of Mutable Data

種別	概要	例
Mutable データ	再取得不可能・困難なもの	ユーザが作成したデータやプログラム アプリケーションの設定ファイル環境
Immutable データ	不変なもの 再取得可能なもの	OS ミドルウェア(データベースなど) アプリケーションなど

3.2 環境提供時のアーカイブ

環境提供時には Mutable データを検出してアーカイブする. 再現環境へ提供されるのはこのアーカイブと, 運用開始時に構築された環境に関する情報, さらに, アーカイブの展開をおこなうプログラムである.

Mutable データの検出方法は運用開始日時を記録しそれ以降に生成・改変されたファイル等をリストアップする手法を用いている. この中には, ユーザ登録情報やシステム・アプリケーションのログ等も含まれる. なお, プロセスファイルやロックファイル, デバイスファイルなどは再現環境で悪影響を及ぼすことが推測されるため, 除外する. このリストを用いて最終手に tarball 化されたものがアーカイブとなり, 構成情報とともに Mutable データとして提供される.

3.3 再現環境の構築

再現環境を構築するためには, 提供された構成情報に基づき基本となるシステムを構築し, Mutable データを展開する. これにより提供者の環境とほぼ同一のシステムが構築される.

Vol.2016-IOT-35 No.13
Vol.2016-SPT-20 No.13
2016/9/24

4. Docker による実装と比較

本研究では実装に Docker[2]を用いる. Docker は Linux コンテナ技術を応用した仮想化基盤である．完全仮想化基盤よりも軽量で，既に Amazon Web Service(AWS)や Google などでも利用されている．筆者らは既に[3]で Docker による実装の一部を行っている．[3]では一つのコンテナ内で実行されるサービスのアーカイブ及び再構築を行っている．ここでは，その結果と併せて，他の提供方法との比較を行った．

4.1 Dockerfile の記述

Docker ではシステムの構成情報をテキスト形式の Dockerfile に記述する．図 3 に典型的な Dockerfile の記述を示す．

```
FROM ubuntu  #  ベースイメージ
RUN apt-get update -y && apt-get dist-upgrade -fy
RUN apt-get install -fy app
EXPOSE 12345  #通信ポートの開放
VOLUME ["/share"]  # ボリュームのマウント
VOLUME ["/archive"]
COPY util.tar.gz
RUN tar xvfz util.tar.gz -C /archive
RUN touch /tmp/ts0
```

図 3　Dockerfile 記述例
Figure 3　Example of Dockerfile Notation of Dockerfile

Dockerfile 内ではまず“FROM”句でベースとなる OS のイメージを指定する．イメージがローカルコンピュータ内に存在しない場合は公式リポジトリである Docker Hub[4]から自動的に取得する．次にその OS イメージ内でパッケージシステムを用いて，ベース OS 内のソフトウェアをアップデートする．次に必要なアプリケーションをインストールし，各種の設定等を行う．”RUN”句を用いることにより，任意のシェルコマンドが実行可能である．これにより，再構築時にはアーカイブを展開して，環境を整備するなどのスクリプトを実行させることが出来る．

また，”COPY”句によって，ホストにあるファイルをコンテナへコピーすることが出来る．上記の例では，アーカイブプログラム及びリストアプログラムを tarball 化しファイルをコンテナ内にコピーして，展開している．

最後の行では touch コマンドでファイルを作成しているが，これは本研究における Mutable データの抽出に用いる利用するタイムスタンプファイルである．アーカイブ時にはこのファイルの日付より新しいものすべてが Mutable データとしてリストアップされる．

この例からわかるように Dockerfile は構成手順がわかりやすいため，追加・変更等が容易であることも Docker の特徴である．

4.2 提供方法の比較

ここでは，提供されたものを利用して環境を構築する上で，OS やアプリケーションのアップデート等が必要である場合を想定し，本手法とソース・データの提供，ディスクイメージの提供とを比較する．提供する環境の OS は Ubuntu 12，アプリケーションは samba を利用したディスク共有を想定している．

（１）　再現の簡易さ

実験環境の再現を考えたとき，被提供者は当該の研究には精通していても必ずしもシステムに詳しいとは限らない．従って，これは他者に環境を提供する上で重要な観点であると考える．表 2 に簡易さを定性的に比較した表を示す．

表 2：　本手法と他の提供方法の比較（簡易さ）
Table 2: Comparison of Proposed Method and other Methods (facility).

手法	簡易さ
本手法	容易
ソース・データ	環境理解，動作環境の調査が必要
ディスクイメージ	アップデートの手動操作が必要

本手法では，Dockerfile 内に必要なパッケージのインストール，RUN による環境構築プログラムの実行を記載することが出来るため，Docker でイメージを構築して，実行することで数回のコマンドで環境を構築することが出来る．

それに対して，ソース・プログラムの提供では，それらを展開する場所，必要なライブラリ等を調査して理解する必要がある．ただし，すでに動作している計算機内にデータ等を展開するため，OS やアプリケーションは既にアップデートされていることも想定できる．

また，ディスクイメージの提供では，イメージの実行自体は Docker と同様簡単であるものの，OS やアプリケーションのアップデートが必要になる．

（２）　再現時間

この項目では，再現する上での時間を比較する．本手法とディスクイメージ提供については，実測したデータを示す．いずれも 5 回の試行による平均値である．なお，ダウンロード，インストール時間は筆者らが用いた環境に依存する．

表 3：　本手法と他の提供方法の比較(時間)
Table 3: Comparison of Proposed Method and other Methods (time).

手法	時間
本手法	6 分 25 秒
ソース・データ	-
ディスクイメージ	20 分 20 秒

本手法における時間は”docker build”コマンドの実行時間であり，ベース OS を Docker Hub から取得する部分も含まれている．ソース・データの提供については調査や環

Vol.2016-IOT-35 No.13
Vol.2016-SPT-20 No.13
2016/9/24

境整備に手作業が多く含まれることが想定されるため対象外としたが，多くの時間が割かれるのは想像に難くない．

また，ディスクイメージの提供は docker の利用イメージを起動させ，コンテナ内でアップグレードコマンドを実行した時間である．

（3） 保存容量

最後に，ホストのディスク利用量の比較を行う．ここでは，samba を想定しているので，ソース・データでは，共有ディスク内の容量 100kB を提供するものとする．

表 3： 本手法と他の提供方法の比較(利用量)
Table 3: Comparison of Proposed Method and other Methods (disk usage).

手法	利用量
本手法	109kB
ソース・データ	100kB
ディスクイメージ	307MB

また，ディスクイメージ提供においては，OS,アプリケーションのアップデート後，数 MB 増加している．

4.3 複数コンテナ環境

さらに本研究では，複数のコンテナからなる環境の再現を行った．環境として用いたのは e-Learning システムとして広く用いられている Moodle[5]である．複数のコンテナを扱うには Docker だけでは操作が煩雑になるため，オーケストレータである Docker Compose を用いた．Docker Composer では構成情報を YAML 形式で記述する．

このシステムは“db”と”moodle”の二つのコンテナで構成

```
db:
  build: ./db
  environment:
    MYSQL_DATABASE: moodle
    MYSQL_PASSWORD: xxxxxx
    MYSQL_USER: moodle
  ports:
  - "3306:3306"
  volumes:
  - "./arc:/arc"
moodle:
  build: ./moodle
  environment:
    MOODLE_URL:
http://hostname:8080
  links:
  - "db:db"
  ports:
  - "8080:80"
  volumes:
  - "./arc:/arc"
```

図 3 Docker Compose の構成情報
Figure 3 Structure Information of Docker Compose

されている．また，moodle の項にの”links”句に書かれている”db:db”は db というコンテナにリンクしており，直接アクセス可能であることを示し，さらに依存関係があることが Docker Compose によって認識される．それぞれのシステム内容は，構成情報の”build”句の右側に書かれているディレクトリ内に Dockerfile が配置されている．この Dockerfile 内に図 2 と同様のシステムごとの構成情報が記載される．

この構成情報を用いて，システムのアーカイブを取得し，moodle のバージョンアップを伴うシステム再構築の実証を行った．しかしながら，moodle が利用するミドルウェア php の依存関係により，moodle だけのバージョンアップは不可能であった．これを打開するため Dockerfile を編集し，開始まではいくつかの手順を踏んだのち，起動を確認することが出来た．

5. 考察

提供方法について比較を行ったが，いずれの観点でも概ね本手法が優位であることが示された．再現の簡易さと再現時間については Dockerfile とユーティリティを用いた自動化によるものが大きい．本研究で実際に用いた Dockerfile は Docker Hub で提供されているものに，タイムスタンプなどわずかな修正の上，用いている．これを考慮すると，Docker 及び Docker Hub による優位性であるともいえる．

しかし，データのアーカイブには多少の注意が必要である．実際には[3]で述べられているように，アプリケーションソフトウェアのバージョンによってファイルの配置が異なるなどの差異があるためである．この障害を緩和する一つの方法は，提供者が利用したバージョンをできるだけ詳細に Dockerfile に記載することである．しかし，この場合，バージョンアップが必要になる場合，被提供者が Dockerfile を書き換えなければならないため負担となることが予測される．また，インストールするアプリケーションが app でバージョンが xx.yy だとすると，Dockerfile 内に

```
RUN apt-get install app-xx.yy
```

のように記述することになるが，ベース OS のバージョンによっては，xx.yy のバージョンがリポジトリに存在しない場合もある．それは実際に 4.3 で述べた環境で起こった．問題は複数コンテナ環境に由来するものではなく，複数アプリケーションによるものである．Moodle の場合は，最新版では php7 での動作を要求しているのに対して，OS(Ubuntu14)でサポートしているバージョンは php5 であった．従って，Moodle をバージョンアップするために，OS, php の二つをバージョンアップするように Dockerfile を修正する必要があった．

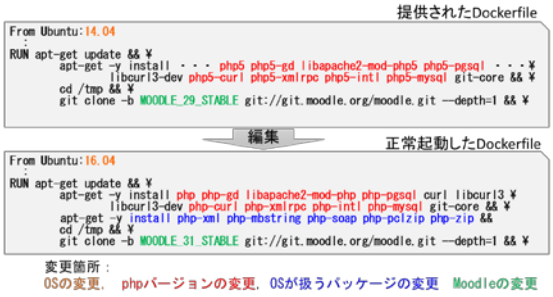

図 4 Dockerfile の編集
Figure 4 Editing of Dockerfile

さらに，php7 ではライブラリのパッケージが変更になっており，一度エラーが出る状態の Moodle を起動し，画面を見ながら要求されているパッケージをインストールすることで，最終的にはサービスが提供された．

これを解消するための方策としては[3]にあるようにバージョン間の差分情報を蓄積した中間データベースが挙げられるが，利用者数が少ない，あるいはライセンスが必要なアプリケーションについては，実現が難しいと思われる．

6. おわりに

本稿では，計算機環境の提供を，更新されたファイルを検出したデータと構成情報を Mutable データとして与える手法を提案し，実行例を他の提供方式と比較した．その結果，ソースコードを提供する方法や，ディスクイメージを提供する方法に比べて三つの観点で優位であることが示された．また，２つのコンテナからなる情報システムを本手法により移行できることを示した．しかしながら複数のアプリケーションのバージョンアップを伴う場合，Dockerfile の記述を被提供者が編集する必要がある場合が生じた．

今後の課題として，Mutable データの世代管理が挙げられる．研究環境は単一の検証だけでなく，提供された環境を用いた発展・改良なども考えられるため，複数世代にわたる環境提供，Mutable データの世代管理も今後取り組むべき課題である．

本研究は，国立情報学研究所の平成 29 年度公募型共同研究テーマ「複数計算機から構成される研究用計算機環境の自動再構築に関する研究」の研究成果である．

参考文献

[1] 横山重俊，政谷好伸，小笠原理，大田達郎，吉岡信和，合田憲人:Overlay Cloud で構成する論文再現環境，情報処理学会研究報告，2015-IOT-31(1), pp.1-8,2015 年 9 月
[2] Docker: http://www.docker.com (2016.8.30)
[3] 石坂徹，桑田喜隆，横山重俊，合田憲人：仮想化基盤上でのシステム再構築の一手法，情報処理学会研究報告 2016-IOT-33(1), pp.1-6,2016 年 3 月
[4] Docker Hub :http://hub.docker.com (2016.8.30)
[5] Moodle: http://www.moodle.org (2016.8.30)