人工知能学会
第18回知識流通ネットワーク研究会
SIG-KSN-018-08

# 研究用の計算機環境の保存と再利用に関する考察

○桑田 喜隆[†1] 石坂 徹[†1] 横山 重俊[†2] 合田 憲人[†2]
[†1] 室蘭工業大学
[†2] 国立情報学研究所

# A study on preservation and restoration of computer environment for scientific research

Yoshitaka Kuwata[†1], Toru Ishizuka[†1], Shigetoshi Yokoyama[†2], and Kento Aida[†2]
[†1] Muroran Institute of Technology, Japan
[†2] National Institute for Informatics, Japan

概要

論文の客観性の確保のため，研究者には研究論文のデータの正確性，及び論文のもととなった実験の再現性が求められる．仮想化技術を使うことで計算機環境を保存し，数年後に環境を再現し，再利用することが容易である．他方，ソフトウェアは日々更新されており，保存されている古い環境を後にそのまま利用することは，コード危殆化及び古いコードに含まれる不具合の観点から望ましくない．本研究では計算機環境を効率よく保存し，目的に応じて環境を再現する方法に関して比較検討を行い，アーカイブの指針を提案する．

Abstract

In order to prove the accuracy of papers, researchers need to preserve detailed methods of their experiments and data used for the experiments. In the field of computer science, computer environments are easy to preserve and re-execute by using virtual machines technologies. However, it is unsuitable to use a set of old software that has been preserved in archives, as it might include defects and vulnerabilities. In this study, we compare methods to preserve computer environments, and propose a guideline for achieving.

## 1. はじめに

論文の客観性の確保のため，研究者には研究論文のデータの正確性，及び論文のもととなった実験の再現性が求められる．計算機を用いた研究の場合には，論文執筆に利用したデータばかりではなく，利用した計算機環境の保存が必要になる．例えば，計算機科学におけるアルゴリズムの検証や，計算機シミュレーション等を用い検証を行う場合には，計算に用いたデータやプログラムを公開するだけでは不十分で，検証を再現するための環境を提供しないと厳密な検証は難しいと．

これに対し，仮想化技術を使うことで計算機環境を保存し，数年後に環境を再現し，再利用することが容易である．更に，仮想マシンイメージとして計算機環境を体系的に保存するアーカイブシステムを構築することも可能である．他方，ソフトウェアは日々更新されており，保存されている古い環境を後にそのまま利用することは，コード危殆化及び古いコードに含まれる不具合の観点から望ましくない．別の手段として，必要なデータを保存しておき，再現時に計算機環境を再構築する方法も考えられる．

そこで，本研究では計算機環境を効率よく保存し，目的に応じて環境を再現する方法に関して比較検討を行う．また，保存方法に関する一般的な指針を提案する．

## 2. 計算機環境保存の目的と利用モデル

本論文では研究者が利用した計算機のハードウェア，基盤ソフトウェア（オペレーティン

[1] Yoshitaka Kuwata
室蘭工業大学
北海道室蘭市水元町２７−１
kuwata@mmm.muroran-it.ac.jp

グシステム，ミドルウェアなど）及び，アプリケーションソフトウェアまで全てを含めて，計算機環境（以下，単に環境）と呼ぶことにする．ただし，計算機のハードウェアとして仮想計算機も含めるものとする．

環境を保存する目的として，本稿では以下のケースを対象とする．

(A) 論文等の証跡
(B) 後に自分で利用する
(C) 他の研究者との共有

(A)の場合は論文に掲載した結果の信憑性を証明するため，論文とともに保存した環境を公開することが想定される．このため，論文執筆時点の環境をそのまま残し，できるだけ正確に再現することが必要である．また，第三者による評価が想定されるため，再現するための環境の一般性や入手性，及び再現可能な期間等が重要になる．

(B)及び(C)の場合は，再現性もさることながら，環境を利用して安定的に研究を継続可能なことが重要になる．例えば，保存後にソフトウェアの不具合が発見された場合には，再現時に修正することが必要になる．また，ソフトウェアの脆弱性が発見されることも想定されるため，再現後も継続してソフトウェアを更新することが必要になる．

図 1 に(A)及び(B),(C)の場合の処理フローのモデルを示す．

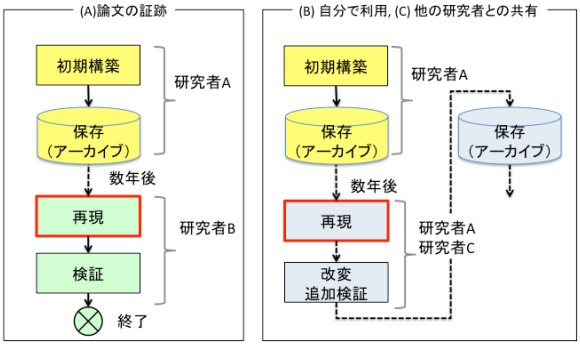


図 1　目的別の処理フローのモデル

(A)のケースでは研究者 A の保存した環境を数年後に研究者 B が再現し一時的に検証を行う．この場合，継続的な利用は想定しない．これに対して，(B)及び(C)のケースでは，研究者 A や別の研究者 C が環境を再現し，新たな研究を行うことを想定している．

## 3. 保存方法の比較のための評価基準

環境の保存方法について論じるための評価の基準として，以下の 7 項目を考慮することを提案する．

(I)　再現内容の再現性
(II)　再現環境の入手性
(III)　保存及び再現作業の容易性
(IV)　保存のために必要な記憶容量
(V)　保存及び再現のための処理時間
(VI)　保存可能な期間
(VII)　再利用性

(I)再現性は，再現した環境で元と同じ計算結果が得られるかどうかを示す．プログラムが同一であったとしても，ライブラリなどの環境の違いで誤差が出ることも考えられる．また，処理時間の測定を伴う場合には，再現環境の性能で結果が変化することが予測される．(II)入手性は同じ環境の入手が容易であるかどうかを示す．商用ソフトウェアを利用した場合，別に入手が必要となる．(III)作業容易性は必要な作業に特殊なスキルが必要であるかどうかを示す．手動の作業を伴う場合，再現環境についての詳細知識が必要になる．(IV)保存のための記憶容量は，少ない方が望ましい．(V)保存及び再現のための処理時間は，短い方が望ましい．(VI)検証可能な期間は，ソフトウェアやハードウェア環境の経年変化により再現が不可能になるまでの期間である．(VII)再利用性は，再現後の改変やアップデートのし易さやを示す指標である．例えば，内部の仕組みが明示的に記述してある場合，再利用性が高い．

2 章で述べた保存の目的ごとに，評価基準の重要度判断の判断基準を表 1 に示す．

表 1　目的と評価基準の重要性（重み係数）

| 評価基準＼目的 | (A)証跡 | (B)自分 | I 共有 |
|---|---|---|---|
| (I) 再現性 | 高(3) | 中(2) | 中(2) |
| (II) 入手性 | 高(3) | 低(1) | 中(2) |
| (III) 作業容易性 | 低(1) | 中(2) | 中(2) |
| (IV) 記憶容量 | 中(2) | 高(3) | 高(2) |
| (V) 処理時間 | 低(1) | 中(2) | 中(3) |
| (VI) 保存期間 | 高(3) | 低(1) | 中(2) |
| (VII) 再利用性 | 低(1) | 高(3) | 高(3) |

## 4. 従来手法とその比較

従来の環境の保存方式として，以下の 3 種類が一般に採用されている．

(1) データ及びソースコードの保存
(2) マシンイメージの保存
(3) 構築方法の保存

それぞれの保存方式の概要及び特徴を以下に示す．

### 4.1 データ及びソースコードの保存方式

プログラムのソースコード及びデータをそのままアーカイブして保存しておく方法である．伝統的な方法としては，ソースコードやデータの置かれたフォルダー配下をまとめてアーカイブする方法が取られる．本方式では，環境全体ではなく，アプリケーションプログラムのみを保存する方法である．再現時に用意した環境上でプログラムは実行される．

例えば，Unix 系の OS で開発したソフトウェアのソースコード配布の場合は，アーカイブに利用するプログラムにちなんで，tarball と呼ばれる形式が利用される．再構築時には，アーカイブ内に含まれるビルド手順（Makefile 等）を利用して，再現時に用意した環境上にプログラムを再構築することが一般的である．

図 2 に本方式を用いた場合の研究環境の保存及び再利用の流れを示す．

近年では，版管理のために Github[1]などのソースコード・レポジトリが利用されており，直接ソースコードを取得することに加え，改変履歴を確認したり，過去の任意の時点のソースコードを取得することが可能である．

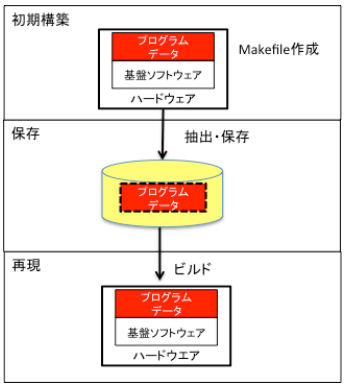


図 2　データ及びソースコードの保存方式

本方式は，以下の利点がある．

- 広く普及している方法で一般性が高い．
- 保存操作が容易である．
- ビルド手順を用意しておくと，再現も比較的容易である．
- 保存のために必要な記憶領域は小さい．
- レポジトリ併用することで，版管理や差分管理を行うことができる．

他方，環境の保存と再利用という観点からは以下の課題がある．

- ソースコードを基にしたソフトウェアの配布に利用される方法であるため，個別の実験で用いた設定条件などは含まれない．このため，別に設定手順書などを用意する必要がある．
- ビルド手順の想定する環境と異なる環境で再現する場合，手動による修正が必要になる場合があり，環境に関する知識やスキルが必要になる．
- 初期構築時に利用した基盤ソフトウェアと異なる環境で再現する場合，修正が必要である．例えばライブラリに依存する場合には，再現時に差異が生じないようにバージョン情報などを明示する必要がある．
- 再現時にコンパイルなどの変換処理が必要になるため，言語処理系などの実行環境に依存する．このため，論文執筆時の環境と完全に同じ結果になるとは限らない．

本保存方式のバリエーションとして，コンパイル済みの実行形式ファイルも合わせて用意しておく方法も考えられる．この場合には，上記の再構築などの問題がなくなる反面，再現環境にバイナリーレベルでの互換性が要求されるため，汎用性が下がると考えられる．

本方式は，プログラムの構成が比較的単純で，外部に依存の少ない場合に有効であると考えられる．

### 4.2 マシンイメージの保存方式

仮想化技術の進展により，計算に利用した環境をマシンイメージとして保存しておくことが容易になっている．近年，構築済みの環境を仮想マシンイメージで配布する例も増えている．特に仮想サーバの提供サービスではクラウド基盤提供者や第三者から提供された構築済

みのマシンイメージを利用する場合も多い. 仮想化ソフトである VMware や Xen, KVM では仮想マシンイメージを保存する方式であり, 他の環境でも再利用可能である. また, 商用のクラウドコンピューティングサービスでも同様にマシンイメージで扱われて, 保存と共用が可能である. 例えば, Amazon Web Services[2]では AMI(Amazon Machine Image)形式でマシンイメージが扱われるが, 許可が与えられていれば, 他人の保存した AMI ファイルをもとに仮想マシンを起動することが可能である.

仮想マシンイメージなどの保存は, そのまま研究に用いた環境の保存に利用することができる. 図 3 にマシンイメージの保存方式の概要を示す.

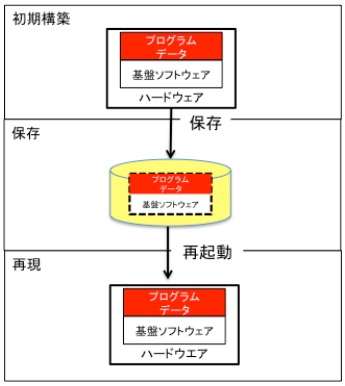


図 3 マシンイメージの保存方式

本方式は, 以下の利点がある.

- 保存のための作業が簡単である.
- 再現時にも特別な作業が不要である.
- OS まで含めて保存時と全く同じ環境が再現時に利用出来る.

他方, 環境の保存と再利用という観点から以下の課題がある.

- ブラックボックス化されており, 別に説明がないと, 変更点や利用方法などがわからない.
- 保存のための記憶領域が大きい.
- 再現時に OS や基盤ソフトウェアなどの環境が古くなっている可能性がある.
- 保存した環境に商用ソフトウェアを含む場合, 再現時にライセンスの取得が必要である. また, アクティベーションやライセンスサーバの設定など, 利用に当たって特別な操作が必要な場合もある.

### 4.3 環境構築手順の保存方式

近年, OS やミドルウェアを含む環境構築の自動化技術が進展している. 例えば, Vagrant[3]や Docker[4]などの自動化ツールを利用すると, 外部レポジトリからソフトウェアを取得し, 起動可能なマシンイメージを作成することが可能である.

例えば, Docker の場合, Dockerfile と呼ばれる構築手順記述様式が利用される. Dockerfile で指定した OS やパッケージを元に, Docker を使って起動可能なマシンイメージを構築する. 外部のソースコード・レポジトリよりソースコードを取得しビルドしたり, 計算に必要なデータを配置することも可能である.

更に Puppet[5]や Chef[6]などの設定自動化ツールを併用すると, より細かなソフトウェア設定も可能である.

研究環境の保存に, 環境構築の自動化技術を応用することが可能である. 図 4 に環境構築手順を保存する方式の概要を示す.

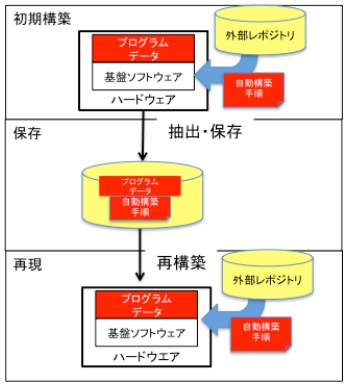


図 4 環境構築手順の保存方式

本方式では, 初期構築時に構築手順のスクリプトを用意しておく. 構築手順においては, 外部のレポジトリからソフトウェアを取得し, 環境を自動構築する. 外部レポジトリのデータをもとに, 手順に従っていつでも環境の構築が可能であるため, 「不変のインフラ(Immutable Infrastructure)」と呼ばれる. 研究者が行った変更のみを保存しておけば, 後に環境を再現することか可能である.

本方式は以下のメリットがある.

- 研究者の行った変更以外は保存が不要であるため, 保存に必要な記憶容量が少ない.
- 構築手順が明示的に扱われるため, 再利用

が容易である.
・最新のソフトウェアを取得して環境を構築する構築手順を作成しておくと,再現時に最新のソフトウェア環境を利用可能である.

他方，以下のデメリットがある.
・構築手順の作成が必要になる.
・環境の再構築処理に時間がかかる.

### 4.4 保存方式の机上評価

前節までに述べた３方式に関して，３章で述べた評価基準をもとに机上の評価を実施した.結果を表２に示す.

表２ 保存方式の机上評価結果（点）

| 方式 / 評価基準 | (1)データ及びソースコードの保存 | (2)マシンイメージ保存 | (3)環境構築手順の保存 |
|---|---|---|---|
| (I) 再現性 | 低(1) | 高(3) | 中(2) |
| (II) 入手性 | 中(2) | 高(3) | 高(3) |
| (III) 作業容易性 | 難(1) | 易(3) | 中(2) |
| (IV) 記憶容量 | 小(1) | 大(3) | 小(1) |
| (V) 処理時間 | 中(2) | 短(1) | 長(3) |
| (VI) 保存期間 | 中(2) | 長(3) | 中(2) |
| (VII) 再利用性 | 高(3) | 低(1) | 高(3) |

表１の重み係数に，表２の評価結果点数を乗じて,目的別の保存方法の合計点数を計算した.

表３ 保存方式の机上評価結果（点）

| 方式＼目的 | (A)証跡 | (B)自分 | I 共有 |
|---|---|---|---|
| (1)データ及びソースコードの保存 | 23 | 24 | 29 |
| (2)マシンイメージ保存 | **38** | **32** | **36** |
| (3)環境構築手順の保存 | 31 | **31** | **38** |

以下に，目的別の机上評価結果をまとめる.

- 保存目的が(A)論文等の証跡として保存する場合，(2)マシンイメージ保存方式が望ましい.
- 保存目的が(B)自己利用のための保存の場合，(2)マシンイメージ保存，または(3)環境構築手順の保存方式が望ましい.
- 保存目的が(C)共有のための保存の場合には，(2)マシンイメージ保存，または(3)環境構築手順の保存方式が望ましい.

全てのケースで，(2)マシンイメージの保存方法は有効である，特に(A)証跡の場合には，再現性，入手性，保存期間などの評価が高い.(A)証跡以外の目的の場合,(3)環境構築手順の保存方式も有効であることが分かる.

以下，本稿では(3)環境構築手順の保存方式に着目する.課題である,構築手順の作成を容易にするため,構築済みのマシンイメージをもとに構築手順の作成支援の仕組みを提案する.

## 5. 構築手順の作成支援手法

環境構築手順を保存する方法をとる場合，研究者が保存と再現を意識して構築手順を作成した上で研究環境を構築することが望ましい.しかし,一般的には研究環境の構築には試行錯誤を伴うため,予め手順化することは難しい場合が多いと考えられる.そこで,既に手動で環境構築が終わり研究に利用した後の環境をもとに，構築手順を作成する方法を検討した.
以下の手順が必要になる.

(a) 変更箇所の抽出
(b) 再現に必要な箇所の抽出
(c) 変更意図の判別
(d) 構築手順の作成

### 5.1 変更箇所の抽出

研究者が作業を開始したマシンイメージと最終的に利用したマシンイメージを比較することで変更箇所が抽出可能である.両マシンイメージをファイル単位で比較することで,変更のあったファイルをリストアップすることができる.また，作業開始時刻が分かっている場合には,ファイルの更新日付を参照することで,リストが作成可能である.

### 5.2 再現に必要な箇所の抽出

(a)の作業で抽出されたファイルは，再現時に不要な変更も含まれる.例えば,環境に依存する設定,キャッシュ,プログラムの実行ログ,排他制御用のファイルなどは再現に必要のないものであるため,リストから除外することが必要になる.環境に依存した項目としては，ネ

ットワーク環境の設定情報，ホスト ID，鍵データ，アカウント情報などが含まれる．

不要なファイルやフォルダーの一覧を作成しておき，リストから除外する方法が考えられる．しかし，上記の配置は OS 等の実装に依存するため，人手によるレビューが必要になる．

### 5.3 変更意図の判別

(a)，(b)の作業の結果，抽出された変更点に意味づけを行う作業が必要になる．

例えば，以下のような変更意図を判別することが必要である．

- 研究者が独自に用意したプログラム
- 外部アーカイブ等から得られるプログラムの変更
- 研究者の用意した実験用データ
- 外部アーカイブ等から得られるデータ
- 実験のためのパラメータ設定
- 中間結果や一時ファイル
- 実験結果

上記の一般的な判別方法はないため，人手による判別が必要になる．

予めファイル配置に関するルールを決めておくことで，簡略化することが可能である．また，判別を省略して，上記全てを研究者の変更として保存しておく方法も考えられる．

### 5.4 構築手順の作成

変更意図に応じて，以下の点に注意し環境構築手順に反映する．

- 外部アーカイブから得られるデータやプログラムは構築時に取得する．
- 研究者が独自に用意したプログラムやデータは研究環境の一部として保存しておく．汎用的に利用可能であれば，外部アーカイブに登録し，環境構築時に取得する
- 中間結果や実験結果は，環境再現に不要なので保存しない．再構築後に確認の際に役に立つので，残しておく選択肢もある．

この作業は，人手で実施することが前提となる．上記の分類に基づき雛形を用意しておくと役に立つと考えられる．

なお，上記の手順の一部を，具体的なシステムの適応した評価結果に関して，文献 7 で報告した．

## 6. まとめと今後の課題

研究用の計算機環境の保存および再利用に注目し，目的に応じて最適な保存方式について机上評価を実施した．また，構築手順の保存方式に注目し，構築手順の作成を自動的に行う方法を提案した．

本研究では計算機環境を単一のハードウェアで実現した場合の保存方法について論じた．近年，複数のハードウェアを利用して分散処理を行う場合が増加している．複数のハードウェアから構成される研究環境の保存および再現に拡張することが課題である．

本研究は，国立情報学研究所の平成 27 年度公募型共同研究テーマ「インタークラウド活用のためのアーカイブおよびリポジトリ管理技術」の研究成果である．

## A. 参考文献

1. GitHub Inc. "GitHub", https://github.com/, Feb, 10, 2016 access.
2. Amazon Web Services, Inc., "Amazon Web Services (AWS) - Cloud Computing Services:", https://aws.amazon.com/jp/, Feb, 10, 2016 access
3. HashiCorp, "Vagrant by HashiCorp:", https://www.vagrantup.com/, Feb, 10, 2016 access
4. Docker, Inc., "Docker - Build, Ship, and Run Any App, Anywhere:", https://www.docker.com/, Feb, 10, 2016 access
5. Puppet Labs, "Puppet Labs: IT Automation Software for System Administrators:", https://puppetlabs.com/, Feb, 10, 2016 access
6. Chef Software, Inc., "Chef | IT automation for speed and awesomeness", https://www.chef.io/chef/ (2016/2/10 access)
7. 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 第 32 回インターネットと運用技術研究会(IOT), 2016 年 3 月 3,4 日