Chef を用いたクラウド分散環境の構築方法

2011SE143 久野 友菜 2011SE228 佐藤 亜衣梨

指導教員: 青山 幹雄

1. はじめに

クラウドの普及によりサーバを短時間で増設することが可能となった. ミドルウェア設定をコードで行う管理自動化ツール Chef は, サーバ増設の人的ミスを軽減できるというメリットから近年注目されている.

本研究では, Chef ユーザの理解支援となるモデル化方法を提案する.

2. 研究の課題

(1) Chef 構成の理解が困難

Chef は構成対象のサーバごとに異なる値を, 個別のスクリプトファイルに分散させるため, 管理内容や対象の把握が困難である. ユーザが Chef の構成を直観的に理解する必要がある.

(2) 再利用が困難

Chef のスクリプトは Ruby ベースで記述するため記法に一貫性が無く, 他者が作成したスクリプトを再利用することが難しい. 記法にルール付けをすることで再利用可能にする必要がある.

3. 関連研究

3.1. Chef

Chef とは, サーバ構築と構成管理のためのフレームワークである[1]. Chef では, 管理されるホストを Node, 設定内容を Recipe, Recipe 外に変数として置かれた Node の属性を Attribute と呼ぶ.

3.2. RDF(Resource Description Framework)

資源(リソース)の情報を機械と共有するための仕組みである. 主語(Subject), 述語(Predicate), 目的語(Object)のトリプルと呼ばれる 3 つの要素で表現する.

3.3. リソースマップ(Resource Map: ReM)

リソースマップにはソースの集合体の構成, 属性, 他のリソースとの関係を記述する[3]. リソースマップは RDF データモデルを用いて表現するが, 明確な構造と制約を持っている.

3.4. Turtle(Terse RDF Trip Language)

RDF のシリアル化フォーマット仕様の 1 つである

3.5. SPARQL(SPARQL Protocol and RDF Query Language)

RDF で記述されたデータにアクセスし, 検索, 操作を行う RDF クエリ言語である.

4. アプローチ

ユーザが目で見て直観的に理解できる Recipe モデルの設計方法を提案し, ユーザの Chef 構築を支援する(図 1).

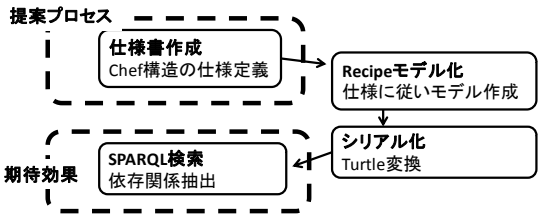


図 1 アプローチ

4.1. Recipe の RDF データモデル仕様書作成

Recipe 構造を RDF データモデルで表現するための基準となるリソースマップの仕様書を定義する. ユーザは仕様書を元に RDF データモデルを作成する.

4.2. アプローチに対する期待効果

Recipe をデータモデル化することにより構造の全貌を一目で把握できるようになる. さらに SPQARQL を用いて依存関係を検索することでユーザの Chef 理解促進がより可能になると期待できる.

5. 提案方法

5.1. 提案プロセスの構成

本研究では Recipe モデル化手法を記した仕様書を提案する. 仕様書では各仕様において, 以下の 2 点を定義する.

(1) 記述定義

Recipe 内の記法ルールを定義する.

(2) リソースマップ定義

RDF データモデルのトリプルと各トリプルの制約条件を定義する.

5.2. 仕様書

本仕様書では以下の 6 つの仕様を定義する.

1) Resource
2) 条件分岐
3) Platform メソッド
4) 配列記法
5) include recipe メソッド
6) Attribute

利用する名前空間は以下の通りである(図 2). 仕様書では図表において挿入例が不変の名前空間は太字で示す.

プリフィックス	対象	挿入例
rec	Recipeに位置する要素	不変
att	Attributeに位置する要素	不変
res	Resource Attributeを指す要素	不変
cookbook_name	Recipeファイル名のリテラル	my_cookbook
recipe_name	Resource種類を指す要素	default, package
platform	プラットフォームの種類を指す要素	不変
attributes	Attributeファイル名のリテラル	不変
chef	Chefの要素	不変
dsl	DSLとして動作が与えられている要素	不変

図 2 名前空間

5.2.1. Resource

(1) Resource 記述定義

Package Resource の例を元に，記述法を定義したものを以下に示す(図 3).

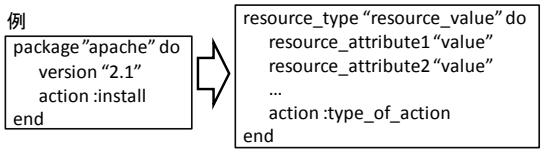


図 3 Resource 記述定義

(2) Resource リソースマップ

記述定義からトリプルとその出現回数を定義したものが以下のリソースマップとなる(図 4).

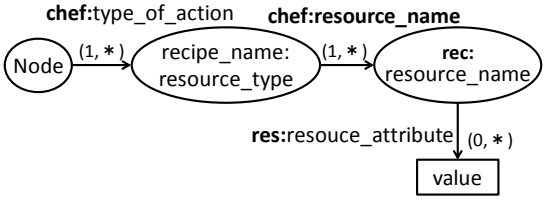


図 4 Resource リソースマップ

5.2.2. 条件分岐

(1) 条件分岐記述定義

if 文の記述法を定義したものを以下に示す(図 5).

```
if terms
条件が真の時に実行する処理
else
条件が偽の時に実行する処理
end
```

図 5 if 文記述定義

(2) 条件分岐リソースマップ

記述定義からトリプルとその出現回数を定義したものが以下のリソースマップとなる(図 6).

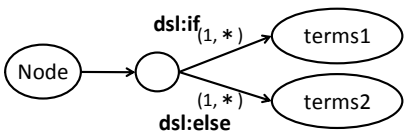


図 6 if 文リソースマップ

5.2.3. Platform メソッド

(1) Platform メソッド記述定義

Platform が Windows の場合の例を元に，記述法を定義したものを以下に示す(図 7).

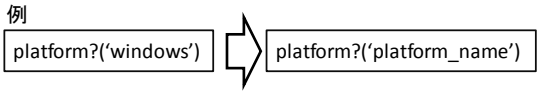


図 7 Platform メソッド記述定義

(2) Platform メソッドリソースマップ

記述定義からトリプルとその出現回数を定義したものが以下のリソースマップとなる(図 8).

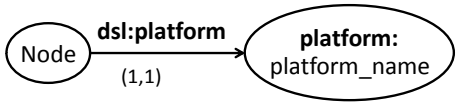


図 8 Platform メソッドリソースマップ

5.2.4. 配列記法

(1) %w 記述定義

配列記法%w の記述定義を以下に示す(図 9).

```
%w{array1 array2 array3}.each do |array_all|

end
```

図 9 %w 記述定義

(2) Resource リソースマップ

記述定義からトリプルとその出現回数を定義したものが以下のリソースマップとなる(図 10).

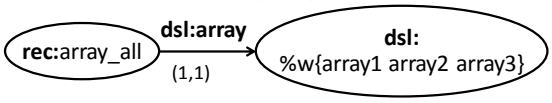


図 10 %w リソースマップ

5.2.5. include_recipe メソッド

(1) include recipe メソッド記述定義

Cookbook"apache"の Recipe"default.rb"をインクルードする例を元に，記述法を定義したものを以下に示す(図 11).

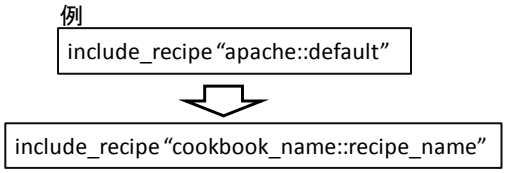


図 11 include recipe メソッド記述定義

(2) Resource リソースマップ

記述定義からトリプルとその出現回数を定義したものが以下のリソースマップとなる(図 12).

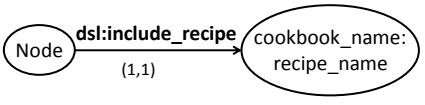


図 12 include recipe メソッドリソースマップ

5.2.6. Attribute

(1) Attribute 記述定義

Attribute に置いた Apache のバージョン情報を呼び出す例を元に, 記述法を定義したものを以下に示す(図 13).

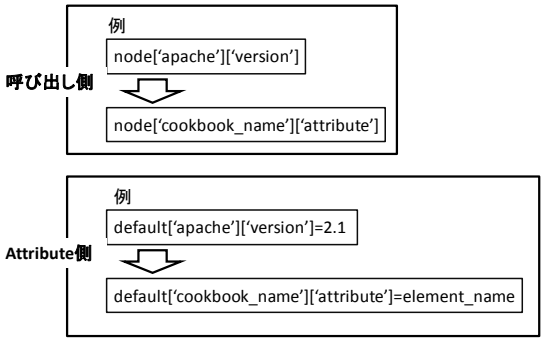


図 13 Attribute 記述定義

(2) Resource リソースマップ

記述定義からトリプルとその出現回数を定義したものが以下のリソースマップとなる(図 14).

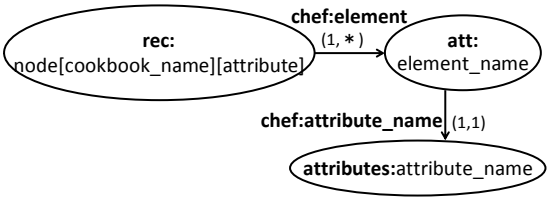


図 14 Attribute リソースマップ

6. 例題による評価

PHP の Recipe を例とした提案の評価を行った. 公式コミュニティサイトにある PHP の Recipe[1]を参考にして RDF モデルの作成をし, Turtle へ変換し, SPARQL 検索が行えるようにした.

6.1. 例題による提案プロセスの評価

PHP の Cookbook を RDF モデルに変換する. 仕様書に従って, Recipe を上から順に RDF モデルに変換し, RDF データモデルが過不足無く作成できることを確認した.

php/recipes/default.rb

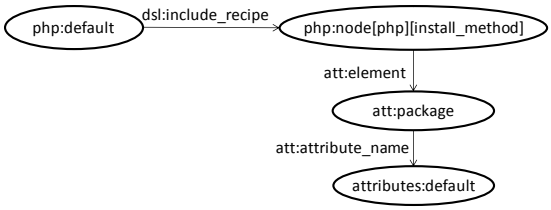
```
include_recipe "php::#{node['php']['install_method']}"

# update the main channels
php_pear_channel 'pear.php.net' do
  action :update
end

php_pear_channel 'pecl.php.net' do
  action :update
end

include_recipe "php::ini"
```

図 15 Cookbook, php の Recipe, default.rb

図 15 の default.rb から作成した 3 つのモデルを図 16 に示す.

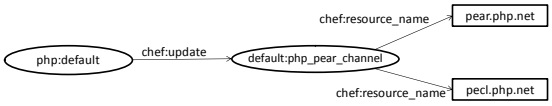
1) Cookbook, php の Recipe, package.rb をインクルード

php の default.rb から他の Recipe を呼び出すコードを RDF で表現した. 他の Recipe の名前は Attribute で記述しており, Attribute を参照すると, package.rb を呼び出している.



2) PEAR チャンネルをアップデート

pear.php.net と pecl.php.net のチャンネルを更新する Resource を RDF で表現した. PEAR と PECL は再利用可能な PHP コンポーネントのためのフレームワークと PHP 拡張のリポジトリである.



3) Cookbook, php の Recipe, ini.rb をインクルード

php の default.rb から他の Recipe を呼び出すコードを RDF で表現した. 呼び出す Recipe は ini.rb である.

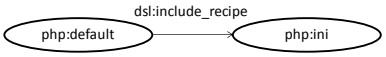


図 16 default.rb の RDF モデル

図 16 の RDF モデルをもとに, RDF/Turtle を作成した(図 17).

```
@prefix att: <http://localhost:3333/att/> .
@prefix res: <http://localhost:3333/res/> .
@prefix package: <http://localhost:3333/package/> .
@prefix ini: <http://localhost:3333/ini/> .
@prefix chef: <http://localhost:3333/chef/> .
@prefix platform: <http://localhost:3333/platform/> .
@prefix rec: <http://localhost:3333/rec/> .
@prefix default: <http://localhost:3333/default/> .
@prefix iis: <http://localhost:3333/iis/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix php: <http://localhost:3333/php/> .
@prefix attributes: <http://localhost:3333/attributes/> .
@prefix dsl: <http://localhost:3333/dsl/> .

att:package att:attribute_name attributes:default .

<http://localhost:3333/php%2Fnode%5Bphp%5D%5Binstall_method%5D> att:element
att:package .

php:default dsl:include_recipe
<http://localhost:3333/php%2Fnode%5Bphp%5D%5Binstall_method%5D> .

default:php_pear_channel chef:resource_name "pear.php.net" , "pecl.php.net" .

php:default chef:update default:php_pear_channel ;
dsl:include_recipe php:ini .
```

図 17 default.rb の RDF/Turtle

6.2. SPARQL による評価

RDF ストア, SPARQL エンドポイントのツールである Virtuoso を利用して評価した.

SPARQL による検索によって，Platform が Windows のときのみの Node の設定内容を抽出できた．CONSTRUCT クエリで検索することによってクエリ結果をトリプルの形で抽出し，視覚的な表現が可能となった．

PHP の Recipe を RDF に変換後，SPARQL クエリで検索した例を以下に示す．

主語が platform:windows であるトリプルを取り出すクエリを用いて検索を行う(図 18)．検索結果は Turtle 形式で出力する(図 19)．図 19 で得たクエリ結果を RDF の視覚化のウェブページを利用し，可視化した結果を以下に示す(図 20)．

```
CONSTRUCT{platform:windows ?p ?o.}
WHERE{platform:windows ?p ?o.}
```

図 18　SPARQL クエリ

```
@prefix dsl: <http://localhost:3333/dsl/>. @prefix platform:
<http://localhost:3333/platform/>. @prefix iis:
<http://localhost:3333/iis/>. platform:windows
dsl:include_recipe iis:mod_cgi. @prefix chef:
<http://localhost:3333/chef/>. @prefix package:
<http://localhost:3333/package/>. platform:windows
chef:install package:windows_package; chef:create
package:cookbook_file, package:template; chef:run
package:execute.
```

図 19　クエリ結果

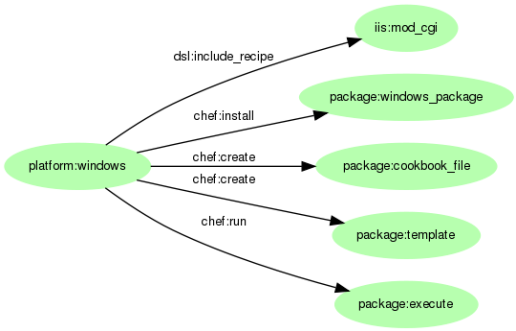


図 20　クエリ結果の可視化

7. 考察

7.1. 提案プロセスに対する考察

仕様書を元に Recipe の RDF データモデルが作成できるようになり，Recipe の視覚的理解が可能となった．要素同士の関連性がトリプルとして表示されることで，第一の課題であった Chef 構成の複雑さも緩和された．第二の課題に対しては，仕様書のルールに基づいてモデル化を行うことで，作成されるモデルに一貫性が生まれ再利用が可能になった．

7.2. 検索に対する考察

従来は Recipe 内のすべての Platform の情報から自身の作業環境に合った値のみを手動で選別していた．Platform を指定して検索をすることで作業環境の固有値，例えばインクルードしている Recipe 名やインストール時に利用している Resource をクエリによって選別可能となった．これにより，ユーザの作業迅速化や，Platform ごとの条件分岐をたどっていく複雑さが軽減できる．

8. 今後の課題

本稿では，Cookbook の RDF モデル作成とその検索のプロセスを提案した．この提案による Chef の視覚的理解の促進を目標としている．Chef を理解し，その後は Recipe の作成の段階となるが，コードでは RDF のような視覚的理解は困難なため，RDF から Recipe が生成されるようなシステムの構築が必要だと考える．

RDF から Recipe への変換方法と Ruby 構文の RDF 作成仕様を考え，ユーザが RDF モデルを作成するだけで，Recipe が生成されるツールの構築が，今後の課題である．

9. まとめ

本稿では，RDF を利用し Chef の理解を促進する方法を提案した．

クラウドが普及し，利用者が増えつつある近年において，サーバなどの構築や増強が間に合わない状況が発生している．そこで，開発者によるサーバ構築管理ツールである Chef の迅速な理解が必要である．Chef は Cookbook 内の依存関係が複雑であり，Cookbook の構成理解と再利用が困難という問題がある．本稿では，仕様書を定義し，そこから Recipe を RDF モデルとして設計することで，Cookbook 構造と依存関係の直観的理解が可能と考えた．また，SPARQL で一部の Tutle 形式の RDF を取り出すことで，ユーザが知りたい部分のみの RDF モデルを確認できる．このプロセスによって，ユーザが Chef を視覚的に理解できる方法を提案した．

参考文献

[1] GitHub, opscode-cookbook/php, 2015, https://github.co m/opscode-cookbooks/php.

[2] C. Lagoze, et al., ORE User Guide - Resource Map Implementation in RDF/XML, 2008, http://www.openarchives.org/ore/1.0/rdfxml [ORE 仕様書 - RDF 構文によるリソースマップの表現，2008, https://www.nii.ac.jp/irp/archive/translation/oai-ore/0.2/rdfsyntax.htm].

[3] 澤登亨彦，樋口大輔，Chef 活用ガイド コードで始める構成管理，KADOKAWA, 2014.

[4] T. Segaran, et al., Programming the Semantic Web, O'Reilly, 2009 [玉川 竜司(訳)，セマンティック Web プログラミング，オライリー・ジャパン，2010].

[5] 吉羽 龍太郎 他，Chef 実践入門 コードによるインフラ構成の自動化，技術評論社，2014.