HPCI 共用ストレージ運用改善のための技術移転

金山 秀智[1)]，近藤　晃[1)]，原田 浩[1)]

平川 学[1)]，芝野 千尋[1)]，尾崎 耕一郎[1)]

1）理化学研究所 計算科学研究機構

hidetomo.kaneyama@riken.jp

概要：AICS の HPCI 運用では、運用改善として作業手順書・スキルマップ管理を実施してきた。さらなる運用品質向上のために構成管理ツール Ansible を導入し、構成管理と運用業務の自動化・機械化を行い、ヒューマンエラーの排除・定型業務の簡素化を実施した。また、Ansible を利用したドキュメント管理と技術移転のためのトレーニングの結果を報告する。

1 はじめに

理化学研究所 計算科学研究機構(以下、AICS)の HPCI 共用ストレージ運用改善のため、定型・定例業務の作業手順書作成、スキルマップ管理などに取り組んできた。これらによって運用に必要な情報が蓄積され、手順の更新と技術移転が進み運用担当者の負担が減少した。さらに運用品質の向上のため、業務の自動化・機械化、ひいては効率的な技術移転を目指して構成管理ツールを導入し、定型業務のトレーニングを試みた。構成管理ツールの選定と導入、導入によって得られた運用品質改善効果と問題点について報告する。

(6) HPCI-SS

項番	内容	作業頻度	近藤	芝野	尾崎	金山	公式関連ドキュメント	ローカルマニュアルの有無
SS-01	Logwatch確認	毎日	○	×	○	主		有
SS-02	Tripwire検知確認	毎日	○	×	○	主		有
SS-03	Zabbix Alertメール確認	毎日	○	×	○	主		無
SS-04	ghpooldigest実行・完了確認(管理表更新)	毎日	○	×	×	主		有
SS-05	共用ストレージ運用部会 議事録作成(要旨・要き起こしまとめ担当)	毎四半期	△	×	○	主		有
SS-06	共用ストレージ運用部会 議事録作成(要き起こし担当)	毎四半期	○	×	○	○		有
SS-07	共用ストレージ運用部会 会議準備?	不定期	×	×	×	主		無
SS-08	富士通 月例報告書のアップロード?	毎月	×	×	×	主		無
SS-09	共用ストレージメンテナンスステジュール作成	毎月	×	×	×	主		無
SS-10	ホスト証明書・サービス証明書更新作業	随時	○	×	×	主		有
SS-11	ローカルアカウント登録作業: ログインノード(esci-wt{01..04})	不定期	○	×	×	主		有
SS-12	ローカルアカウント登録作業: ログインノード・「京」ユーザ向け-(esci-wt{05..10})	不定期	○	×	×	主		有
SS-13	ローカルアカウント削除作業: ログインノード(esci-wt{01..10})	不定期	○	×	×	主		有
SS-14	ローカル*テスト*アカウント登録作業: ログインノード(esci-wt{01..10})	不定期	×	×	×	主		無
SS-15	Gfarmアップデート	毎月	○	×	×	主		有
SS-16	消失・破損ファイル確認	不定期	○	×	×	主		有
SS-17	ログインノードのデフォルトシェルの変更	不定期	△	×	△	主		無
SS-18	調達検討会議 月例報告資料作成	毎月	○	×	×	主		無
SS-19	調達検討会議 議事メモ作成	不定期	○	×	×	主		無
SS-20	調達検討会議準備?	不定期	△	×	×	主		無
SS-21	ヘルプデスク対応(Gfarm・共用ストレージ関連)?	不定期	○	×	×	○		無
SS-22	共用ストレージ マニュアル修正?	不定期	○	×	△	○		無
SS-23	DDNストレージログの確認?	毎日	○	×	×	主		無
SS-24	共用ストレージ案件確認?	不定期	○	×	×	主		無
SS-25	ユーザプロセス停止	不定期	×	×	×	主		無
SS-26	Zabbix設定変更?	不定期	×	×	×	主		無
SS-27	共用ストレージメンテナンスアナウンス	不定期	○	×	△	主	共通ドキュメント参照	無
SS-28	HW故障対応?	不定期	○	×	×	主		有
SS-29	起動・停止?	不定期	○	×	×	主		有
SS-30	管理者アカウント登録作業	不定期	○	×	×	主		無

図 1: スキルマップ

2 構成管理ツールの導入

AICS の HPCI 共用ストレージはメタデータサーバ 2 台、ファイルサーバ 16 台、ログインノード 10 台で構成されているため、複数のホストに対しての更新を、正しく・短時間で実施する必要がある。また、メンテナンス等で実施するプロセス停止・起動、動作確認等の定型業務についても、短時間で実施しサービスのダウンタイムを短くすることが求められる。

HPCI 共用ストレージの運用にあたり、定型・定例業務、障害対応に関してローカルドキュメントとしての作業手順書、障害対応指針などを作成し、運用品質改善をはかってきた。さらに、作業手順書の操作内容を自動化・機械化することにより、ヒューマンエラーによる誤操作の排除、定型業務の実施手順の削減を目的として構成管理ツールを導入することとした。

導入にあたっては、以下の観点から Ansible を採用し、運用チーム内での利用促進、技術移転に努める事とした。

- 既存の運用向けスクリプトが再利用可能
- 管理対象へのパッケージ設定・追加が不要
- 導入に追加サーバ等の資源が不要
- ネットワーク設定・構成変更が不要
- プッシュベースで利用可能
- 記述形式が YAML であるため、コード可読性が高い
- ドキュメントが充実している

・　構成管理に加えて、プロセスの起動・停止、動作確認等の運用業務に利用可能

特に選定にあたっては、HPCI 共用ストレージがすでに実運用段階にあり 70 課題に利用されている現状を鑑み、以下を重要視した。

・　既存の運用向けスクリプトが再利用可能
・　管理対象へのパッケージ設定・追加が不要
・　導入に追加サーバ等の資源が不要
・　ネットワーク設定・構成変更が不要

我々は Ansible の導入として HPCI 共用ストレージサービスの起動と停止及びその確認等の定型業務を自動化し、チーム内メンバに技術移転を目的としたトレーニングを実施し構成管理ツールの導入効果を検証した。

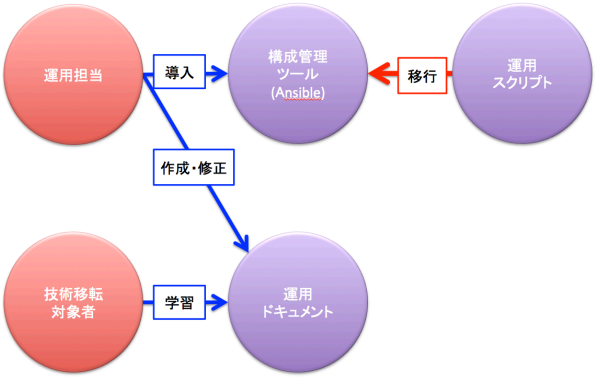

図　2: 構成管理ツール Ansible 導入

3 HPCI 共用ストレージサービス定型業務の自動化

HPCI 共用ストレージ関連サーバのパッケージや設定ファイルの構成管理に加え、次のような定型業務を Ansible で自動化した。

・　サービスの起動・停止
・　サービスの動作確認
・　ReadOnly 設定/解除
・　motd によるメンテナンスアナウンス

例えば、サービスの起動・停止の場合、自動化の実装にあたっては、Ansible の Playbook や Role 等の構成記述ファイルに以下のような記述を行い、デーモンプロセスの起動・停止だけでなく、その確認作業を自動化する実装とした。

これにより、確実なサービスの起動・停止を Ansible のコマンド起動だけで実施できるようになり、実施手順が削減された。

```
(snip)
- shell: "{{ GFSD_INIT_FILE }} status >/dev/null;echo $?"
register: result

- shell: "ps -ef | grep [/]usr/local/gfarm/sbin/gfsd | grep
{{ GFSD_HOSTNAME }} >/dev/null;echo $?"
register: ps_result
when: result.stdout == '1'

- shell: "killall gfsd"
  ignore_errors: yes
  when: ps_result.stdout == '0'
  register: killall_result

-  fail: msg="プロセス(gfsd)のkillallの実行に失敗しました"
  when: (ps_result.stdout == '0') and (killall_result|failed)
(snip)
```

図　3: gfsd 起動確認

4 ドキュメントの二重管理の解消

実施手順が削減され、運用品質の向上に寄与した事は確かであるが、既存の作業手順書に加えて Ansible による実施手順の解説も必要となり、実施手順に関するドキュメントの二重化が発生した。構成管理または定型作業を自動化した 16 項目に加えて、今後も増加する自動化項目のドキュメントを全て二重化して管理するのは困難である。また、技術移転では、複数のドキュメントを参照することとなり、受講者に負担を強いることになる。このため、Ansible の Playbook に作業内容を直接記載し、構成記述と構成管理の実施動作を 1 つのドキュメントに統合することとした。

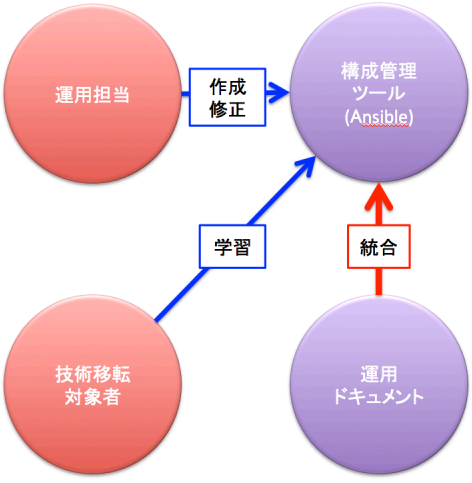

図　4: 運用ドキュメント統合

```
- hosts: FILESERVER
  sudo: yes
  roles:
    - { role: gfsd-check, GFSD_PS_STATE: False }
    - { role: gfsd-start }
    - { role: gfsd-check, GFSD_PS_STATE: True }
```

図 5: 統合前の Playbook

```
#OVV# gfsdを起動する。
#SFC# Gfarmのストレージサーバにおいて、gfsdデーモンを起動する。
#SFC# 前提条件: ストレージサーバにおいて、ディスク領域がマウントされていること。
#SFC#          : ストレージサーバにおいて、gfsdが起動していないこと。
#SFC# 実行対象: Gfarmのストレージサーバ
#SFC# 入力     : 無し。
#SFC# 結果     : gfsdデーモン起動の成功/失敗。
#SFC# 結果確認: /etc/init.d/gfsd statusコマンドによるgfsdの起動確認。
#DCV# GFSD_INIT_FILE: 起動ファイル名のフルパス
#DCV# GFSD_PS_STATE: 状態確認時のステータス判定用のフラグ(False/Trueで停止/起動を切り替え)
- hosts: FILESERVER
  sudo: yes
  vars:
  roles:
    - { role: shared-storage/server/gfsd/common/set-fact, GFSD_INIT_FILE: /etc/init.d/gfsd }
    - { role: shared-storage/server/gfsd/common/check/mount }
    - { role: shared-storage/server/gfsd/service-ps/check, GFSD_INIT_FILE: /etc/init.d/gfsd, GFSD_PS_STATE: False }
    - { role: shared-storage/server/gfsd/service-ps/start, GFSD_INIT_FILE: /etc/init.d/gfsd }
    - { role: shared-storage/server/gfsd/service-ps/check, GFSD_INIT_FILE: /etc/init.d/gfsd, GFSD_PS_STATE: True }
```

図 6: 統合後の Playbook

ドキュメントの統合は、Ansible の Playbook と Role に、「#」と 3 文字キーワードのコメントを記載することで実現した。コメントを記載した Playbook に対してドキュメント生成スクリプトを実行することで、ドキュメントが出力される仕組みである。例えば Playbook に「#OVV」や「#SFC」のキーワードと説明文を記載しておくと、「概要」「仕様」として出力される。図 6 からドキュメントを生成したのが図 7 である。

ドキュメントの追加をするにあたり、以下の 3 点を併せて実施した。

・Ansible コードのリファクタリング

・Playbook、Role、ディレクトリの命名規則制定

・ディレクトリ階層数の制限規則制定

いずれも技術移転時のコードの難読化を防ぎ学習効果を上げるためである。

Ansible にドキュメントを統合し二重化を解消したことで管理コストは減少した。また、更新作業においてドキュメントとコードの更新がセットになることで、ドキュメントとコードの不一致が発生しなくなった。

```
% ./bin/ansibledoc.sh -f shared-storage_server-gfsd_service-ps_start.yml
#Ansible-Playbook: shared-storage_server-gfsd_service-ps_start.yml
====== 概要 ======
gfsdを起動する。
====== 仕様 ======
Gfarmのストレージサーバにおいて、gfsdデーモンを起動する。
前提条件: ストレージサーバにおいて、ディスク領域がマウントされていること。
        : ストレージサーバにおいて、gfsdが起動していないこと。
実行対象: Gfarmのストレージサーバ
入力    : 無し。
結果    : gfsdデーモン起動の成功/失敗。
結果確認: /etc/init.d/gfsd statusコマンドによるgfsdの起動確認。
====== 変数 ======
GFSD_INIT_FILE: 起動ファイル名のフルパス
GFSD_PS_STATE: 状態確認時のステータス判定用のフラグ(False/Trueで停止/起動を切り替え)
====== 実行内容(詳細) ======
1. gfsdのinitスクリプトから、gfsdのオプションを取得し、gfsdに設定されているカノニカルなホスト名を取得
1.1.  initファイルからgfsdのオプションを取得
    $ cat {{ GFSD_INIT_FILE }} | grep esci-wgfs | egrep "/gf[0-9]
(snip)
6. gfsdのプロセスをチェックする
6.1.  プロセスが存在するか確認
    $ {{ GFSD_INIT_FILE }} status >/dev/null;echo $?
(snip)
```

図 7: 生成したドキュメント

5 技術移転トレーニング

HPCI 共用ストレージの運用は、高負荷かつ他機関との協調作業が多く発生するため、複数の運用メンバを確保する必要がある。このため、技術移転トレーニングを行い、運用担当者以外の作業員へ HPCI 共用ストレージの定型業務の実施手順や運用ノウハウを身に着けることとした

2016 年 8 月に、Ansible を利用した第 1 回目のトレーニングを実施した。トレーニング項目は、メンテナンス定例業務として実施する以下の 3 項目である。

- gfsd プロセスの起動・停止
- パッケージと Gfarm のアップデート
- HPCI 共用ストレージサービスの動作確認

トレーニングは以下の手順で実施した。

1. トレーニング項目のドキュメント確認
2. 不明点を運用担当者に質問
3. テスト環境での Ansible 実行・結果確認
4. 本番環境での Ansible 実行・結果確認

本番環境での作業を確実なものにするため、また自動化されている内容を理解してもらうため、Ansible 実行前に可能な限り疑問を解決する手順とした。

今回のトレーニングでは、以下の効果があった。

・ トレーニング受講者からの質問内容が具体的
・ 本番環境でのトレーニング時間が短縮
・ トレーニング受講者からの改善案(Ansible のコード/ドキュメント双方)の提示

今後、トレーニング受講者からの意見により浮かび上がった以下の問題点について改善を試み、第 2 回のトレーニングでは効果を確認する予定である。

・ 全てのエラー処理が網羅されておらず、対処手順が自動化・ドキュメント化されていない箇所がある
・ Ansible 実行時にしか読み込まれない変数があるため、ドキュメント生成時に変数が展開されない

6 まとめ

HPCI 共用ストレージの運用品質向上のために、構成管理ツールとして Ansible を導入し、実際の運用業務に用いた。さらに技術移転のためのトレーニングも実施した。構成管理に加え、サービスの起動・停止だけでなく確認作業も自動化されるなど、品質改善効果がみられたが、手順書の二重化など運用コストが増大する要素も見受けられた。ドキュメントの二重化に対しては、構成記述ファイルに実施手順内容を盛り込むなどの工夫を行い、ドキュメントを統一して運用している。

今後の課題としては、アカウント管理や障害対応などのより複雑で実践的な運用業務に対しても構成管理ツールを導入し改善効果を検証する必要があげられる。

また積極的に技術移転トレーニングを実施し、さらなる品質向上、要員確保につなげるべく構成管理ツール自体の運用改善が求められる。

特に障害対応については、技術移転や自動化が後回しになっていたが、運用ドキュメントの充実により現象の切り分け・対処方法の手順化が進んだため、今後は Ansible による自動化と技術移転トレーニングを進める予定である。