「富岳」の運用を支えるOSS活用事例

PCCC AI/HPC OSS活用ワークショップ

2024年2月5日

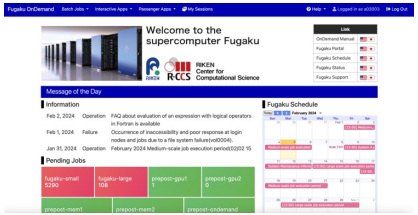
理化学研究所 計算科学研究センター
山本 啓二

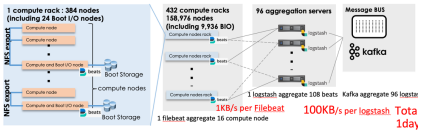
OSSの活用

- ある機能を実現するときの選択肢
  - 独自開発
  - 既存のソフトウェアを利用
    - OSS: 自由に拡張できる
    - プロプライエタリ/SaaS: OSSより機能・サポート面で優位な傾向がある
- 「富岳」では多数のOSSを選択し利用している
  - 同じ機能を実現するOSSにもいくつか選択肢があるため、比較検討が必要

「富岳」のソフトウェア

- HPCの基本的な環境(ファイルシステム・スケジューラ・並列環境・運用管理)は富士通製TCS(プロプライエタリ)を利用
- TCSのみでは運用が難しい部分は独自開発するよりも既存のOSSを活用
  - 利用者ポータル, ドキュメント, ダッシュボード, 認証システム
    - Web系
  - デプロイ, 運用手順, 監視/ログ収集, リポジトリ
    - 運用系
  - 運用分析/可視化, ストリーミング処理, データベース
    - 分析系





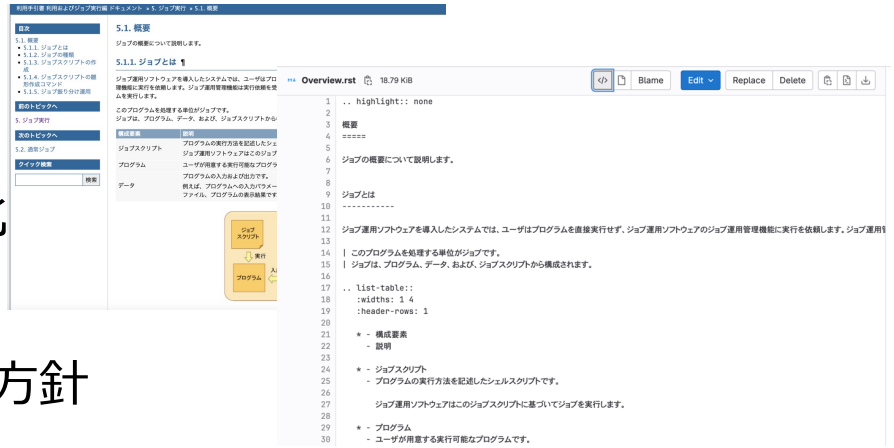
利用者ポータル (ウェブ系)

- 利用者向けに運用情報やドキュメントを提供
- 「京」のとき
  - 独自のテンプレートエンジンを利用した手組みのhtml
  - Perlでのcgi
- 「富岳」になって
  - 管理コストの低減のためにCMSを導入
    - WordPressとDrupalを比較し、デフォルトで多言語に対応しているDrupalを選択
    - たぶんシェアはWordPressの方が多い？
  - cgi部分はウェブAPIフレームワークを導入
    - Perl, php, python, go => python
    - Django, Flask, FastAPI => FastAPI
    - フレームワークの学習コストが小さく軽量なOSSを選択


```python
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
def read_root():
    return {"Hello": "World"}


@app.get("/items/{item_id}")
def read_item(item_id: int, q: str = None):
    return {"item_id": item_id, "q": q}
```

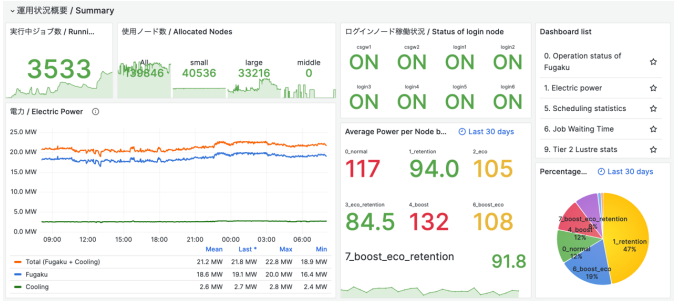
ドキュメント (ウェブ系)

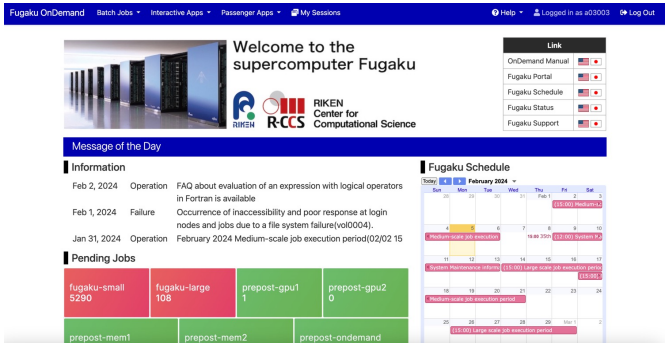
- 利用者向けドキュメントの作成
- 「京」のとき
  - Wordで書いた利用マニュアルをPDF化
- 「富岳」になって
  - オンラインマニュアルを主としPDFも残す方針
  - ドキュメント作成OSS: sphinx, mkdocs, asciidoc
    - どれも軽量のマークアップ言語でドキュメントを記述するといい感じにレイアウトしてくれる
    - デフォルトでPDF化できるのはsphinxのため、「富岳」のドキュメントはsphinxを選択
    - mkdocsはmarkdownで書けるため、オンラインのみであればmkdocsを選択していたかも
  - Gitリポジトリでドキュメント管理をできるようにもなった



ダッシュボード (ウェブ系)

- 利用者向けに運用状況を可視化
- 「京」のとき
  - RRDtoolでジョブ数/利用率をグラフ化し利用者ポータルに掲載
- 「富岳」になって
  - Javascriptのライブラリで高度なグラフ生成ができるため、静的なグラフ画像生成は廃止 (plotly, Google chart)
  - ダッシュボードとしてGrafanaを選択
    - ダッシュボードを作るなら、これしかない
    - Gangliaは用途が監視から可視化まで含まれており、自由にダッシュボードを作りたい場合には向かない





認証システム (ウェブ系)

- 利用者の認証方法
- 「京」のとき
  - 認証が必要なのはSSHおよび利用者ポータルのみ
  - 利用者ポータルは証明書認証
- 「富岳」になって
  - 利用者ポータルの他にWebAPI, OpenOnDemandなど認証が必要なサービスが増加
    - 特にWebAPIには標準の認証方法OpenIDConnectが求められる
  - セキュリティ的に独自実装やライブラリの利用よりも、よく利用されているOSSのIDPであるkeycloakを選択
    - OSSではkeycloakが最適、ただし運用負荷はすこし高い
      - SaaSのAuth0なども最新の認証(Passkey)に追随しているのでよさそう

Jupyter/HPCインターフェイス (ウェブ系)

- 利用者がインタラクティブな操作が可能なインターフェイスを提供
- 「京」のとき
  - なし
- 「富岳」になって
  - OpenOnDemandを導入　(独自実装を試みた…)
    - 認証にはKeycloakを利用
  - OpenOnDemand内にポータル機能の移植を検討
    - たとえば公開鍵の登録画面

運用手順 (運用)

- Infrastructure as Code
  - 手順書やパラメータシートを利用した手動でのプロセスの代わりに、
    コードを利用して構成管理・プロビジョニングを管理
  - コマンドを羅列したシェルスクリプトでもできるが、専用のフレームワークがある
    - Ansible （富岳で利用), Terraform (クラウドプロビジョニング用), Chef, Puppet
    - Ansibleの設定ファイル(レシピ)は yaml なのでgitでバージョン管理も可能
- 障害対応時や定常的な操作には手順書もある
  - LC4RI (Literate Computing for Reproducible Infrastructure)
    - Jupyter notebookを使って手順書、実行コード、実行結果を管理
      - JupyterからAnsibleを起動など

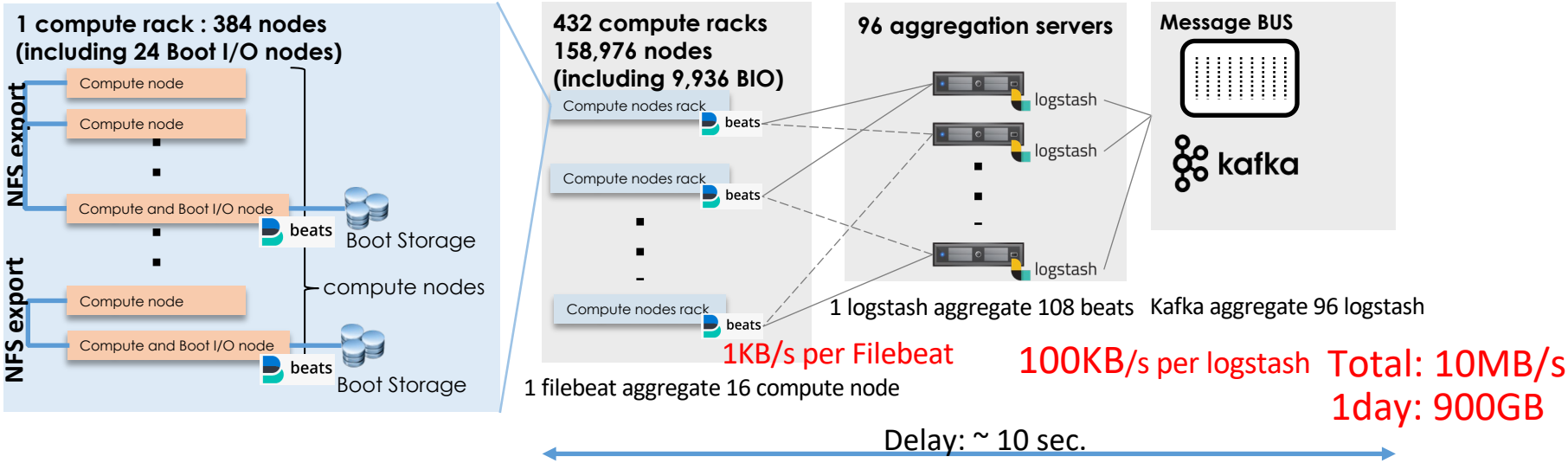
監視システム (運用)

- 代表的な監視システム
  - OSS: Nagios, ZABBIX, prometheus
  - SaaS: Mackerel, Datadog
- 「京」では運用初期にはNagiosを、後期にはZABBIX, Prometheusを導入
  - 多数のアラートの管理が難しい (Nagios)
  - 設定がウェブUIから (IoCではない)/メトリクスが増えると性能低下 (ZABBIX)
- 「富岳」ではprometheusを導入
  - OpenMetrics: メトリクスのデータ定義の標準
  - 計算ノード以外のサーバ全てでnode_exporterを稼働
    - luster_exporter, tcs_exporter(富士通製スケジューラ用のエクスポータ)
    - Grafanaによる可視化も容易に実現
  - アラートに対応した手順を(基本的には)LC4RIで用意
    - 300件以上のアラート登録
  - バックエンドDBはPrometheus, VictriaMetrics, InfluxDB

ログ収集 (運用)

- 管理対象のノード数が増えると、それぞれのノードに入ってログを確認する手間が増える
  - ログ管理ノードにログを集めて、そこで集中的に管理/監視する
    - grepは大変なのでfull-textデータベース化、ストリーミング処理
- 代表的なログ収集ソフトウェア
  - OSS: rsyslog, fluentd, logstash (elastic stack)
  - SaaS: Datadog
- 「京」では運用中盤からfluentdを導入
  - 性能面および管理面でも問題なく稼働
- 「富岳」ではログの全文検索としてElasticsearchを導入したことで、ログ収集ソフトウェアもLogstash, filebeatとelastic stackを利用
  - どちらを使うかは好みによる

HPC Platform: Log & Metrics collection from Compute nodes



Boot storage is shared with compute and I/O node by NFS

Logs of each node are stored boot storage

beats, logstash and Elasticsearch are Elastic Stack: https://www.elastic.co/

Filebeat is a lightweight log shipper written in Go language. Its function is limited to log forwarding only.

Logstash has data transformation and analysis function in addition forwarding. It runs on Java and has a large memory footprint.

運用分析/可視化 (分析)

- ダッシュボードを見ると大抵のことが把握できるが、より深掘りしたいときにはダッシュボードの参照先DBを直接見たい
- 多数のデータソースと連携して、データ分析・可視化できるBIツールがある
  - OSS: Redash, Apache superset, Metabase
  - データソース: Excel, CSV, 主要なデータベース
- 「京」ではRedashを利用
  - 100以上の分析クエリを蓄積/共有
- 「富岳」でも引き続きRedashを利用
  - ただし、活用にはデータが整理されてデータベースに入っていることが重要

理研で開発中のOSS

- Fugaku WebAPI
  - 「富岳」の操作(主にジョブ) をウェブAPIから可能としたもの: 実サービス中
  - 実装で利用したOSS: FastAPI, Keycloak
  - WebAPI自体もgithubで公開
  - 現在Webhookの実装を検討中
- Lens3
  - ローカルのファイルシステムをS3として見せるソフトウェア
  - 利用例: 利用者が自身のホーム/データディレクトリをS3のバケットとして公開できる
    - 読み書きはユーザ自身の権限を利用
  - クローズドでサービス中: 来年度には利用者に公開