Vol.2021-HPC-178 No.14
2021/3/15

センサー情報を意識したジョブスケジューリング実現のための標準ジョブ履歴スキーマの提案

野村 哲弘[1]　滝澤 真一朗[2]　三浦 信一[3]　遠藤 敏夫[1]　松葉 浩也[4]

概要：電力・温度などのセンサー情報を活用したジョブスケジューリングはデータセンター全体の省電力・効率運用のためにその必要性が年々高まっている．しかしながら，各データセンターで取得しているセンサー・ジョブ履歴情報の粒度や形式はサイトや収集項目・収集手段ごとに異なり，多数のセンサー情報を横断的に解析するスケジューラの実装は難しい．本報告では，このようなジョブスケジューリング実現に必要なセンサー情報の粒度について議論し，システム全体にわたるデータを活用したスケジューリングに用いるセンサー情報を付加したジョブ実行履歴のフォーマットと，その変換方法について提案する．

1. はじめに

スーパーコンピュータやクラウドにおけるジョブスケジューラは，多数のユーザによる多量のタスクを公平に処理するために必要不可欠なコンポーネントである．スケジューラの目的関数として従来からジョブの充填率や，ターンアラウンドタイムなどの指標が用いられてきたが，近年では計算資源量に加えて消費電力が制約要因となるなど，ジョブスケジューリングにおける制約や目的関数が多様化している．このような環境下において，様々なスケジューリングの研究や，各データセンターにおける実践が行われてきているが，それぞれの環境における独自のセンサー情報やそのフォーマットに依拠しており，その成果を他で活用することが難しいという問題がある．

本報告では，主に電力制約などを意識したジョブスケジューリングの実現と検証のために必要となるジョブの履歴情報の性質について考察し，それらを包含できる標準ジョブ履歴スキーマを提案する．また，既存のデータセンターにおいて取得しているセンサー情報とスケジューラの情報を組み合わせることで，提案ジョブ履歴スキーマに合致するデータが取得できることを示す．このような標準スキーマを利用することで，ジョブスケジューリングについての研究開発の成果の共有や活用を容易にすることが期待できる．

なお，スーパーコンピュータにおけるジョブ実行ログの解析もしくは公開を行うためには，利用規約や個人情報保護規定等の整備が必要となるが，本報告はデータの取得方法および加工方法の技術面についてのみ言及することとし，これらの制度面に関する議論は行わない．

[1] 東京工業大学
[2] 国立研究開発法人産業技術総合研究所
[3] 国立研究開発法人理化学研究所
[4] 東京大学

2. 背景

2.1 既存の並列ジョブの実行ログフォーマット

既存のスーパーコンピュータにおける標準的なログフォーマットとして，Parallel Workload Archive [1, 2] が利用している SWF 形式 [3] および，Grid Workload Archives [4] が利用している GWF 形式が挙げられる．SWF 形式は CSV(カンマ区切りテキスト)，GWF 形式は CSV もしくは SQLite データベースの形式で格納され，スペース効率性とアプリケーションからの利用の簡便性を両立させており，JSSPP Workshop におけるワークロードアーカイブ [5] や，Alea Simulator [6, 7] などでも採用されているデファクトスタンダードといえるログ形式である．しかしながら，これらのログフォーマットが作られたのは 1990 年台と古く，巨大なメモリ共有型並列計算機環境を前提に並列数は必要な CPU コアの数で記述する形となっている．このような 1 次元の並列度表現では，今日のクラスタ型計算機では当たり前となっているノード内並列とノード間並列の区別や，GPU に代表されるアクセラレータに関する記載ができず，並列数が 1 ノード当たりのコア数を超えている場合にマルチノードのジョブとするなどの何らかの暗黙的仮定をおかない限り，今日のスーパーコンピュータにおける並列性の利用状況を反映させることが出来ない．近年では，東京工業大学におけるスーパーコンピュータ TSUBAME [8] や産業技術総合研究所における AI 橋渡しクラウド (AI Bridging

Cloud Infrastructure，ABCI) [9, 10] などではノード内の資源分割を積極的に取り入れることで，同時に実行できるジョブ数の増加を見込んでおり，先述のような暗黙的仮定すら成り立たなくなってきている．また，データベースとして用いられている CSV および SQLite では，特に前者において項目名の拡張に弱く，並列性の情報や，本報告で目的としている消費電力などのセンサー情報を付加することが難しい．

2.2 既存のセンサー情報のフォーマット

スーパーコンピュータを構成する計算機や電力・冷却設備が生成するセンサー情報については，生成源が膨大にあることから 1 箇所あるいは少数の箇所に集約して，データベース化して管理することが多い．その場合のセンサー情報のフォーマットは，2 階層で考える必要がある．1 階層目は，情報生成源のセンサーからデータベースに送信する情報のフォーマットである．2 階層目は，人やアプリケーションプログラムが情報をデータベースから取得し利用する際のフォーマットである．いずれの階層も従来はセンサーやデータベース毎に異なるフォーマットが用いられていたが，1 階層目については，近年 Redfish や OpenMetrics などの API，フォーマットの標準化が進んでいる．Redfish [11] はサーバ管理インタフェースであり，RESTfull API による操作や監視，JSON によるデータ表現を定義している．OpenMetrics [12] はセンサーなどの情報生成源が http プロトコルでテキストベースフォーマットのデータを提供することが規定されている．2 階層目については，我々の知る限り，共通フォーマットとして定義され，広く利用されているものはない．データベースソフトウェア毎に異なるデータフォーマットを有しているが，最終的に人やアプリケーションが利用する際には，列にタイムスタンプと項目ごとの観測値を持つ CSV 形式に変換されて利用されることが多い．

2.3 節に示す通り，既存のデータセンターの多くではセンサー情報はデータベースに集約して蓄積している．そのため，本研究では 2 階層目のフォーマットを対象とした議論，及び提案を行う．

2.3 データセンターにおけるモニタリングツール

データセンターにおける各種情報の収集は，基本的には既存のモニタリングツールに頼ることとし，必要な項目が計測できていない部分に関してのみ追加の設定もしくはソフトウェアの導入を行うべきである．なぜなら，モニタリング情報の取得および蓄積にも，計算資源およびストレージ資源を必要とし，またセンサーの種類によっては複数のプログラムから同時にデータを要求されることを想定していないものもあるためである．以下，TSUBAME3 および ABCI の事例をもとに，実際に使われているモニタリングツールについて説明する．

2.3.1 スーパーコンピュータ TSUBAME3 での事例

東京工業大学が運用する TSUBAME3 [8] では，それぞれ 2 CPU(計 28 物理コア), 2 GPU からなる 540 台のノードについて，最小で 1 物理コア, 0 GPU の単位まで階層的に資源分割し，各部分で別のジョブを同時に実行するスケジューリングを行い，実効的なノード利用率の向上を図っている．

ジョブスケジューラは Univa Grid Engine(現: Altair Grid Engine) [13] を用いており，ジョブの実行履歴は qacct コマンドで参照できる標準のアカウンティングデータベースおよび，通常のスケジューラログに加えて，課金管理用に利用者管理ポータルにおける SQL データベースに情報を格納している．スケジューラの挙動という面においては，ジョブスケジューラのアカウンティングデータベースの内容が他のログにおける記録項目を包含しており，ジョブの投入ユーザ，投入時刻，開始時刻，終了時刻，使用ノードおよび使用 CPU コア/GPU 情報などが格納されている．

図 1 に TSUBAME3 におけるジョブ情報の蓄積に関するシステム構成を示す．

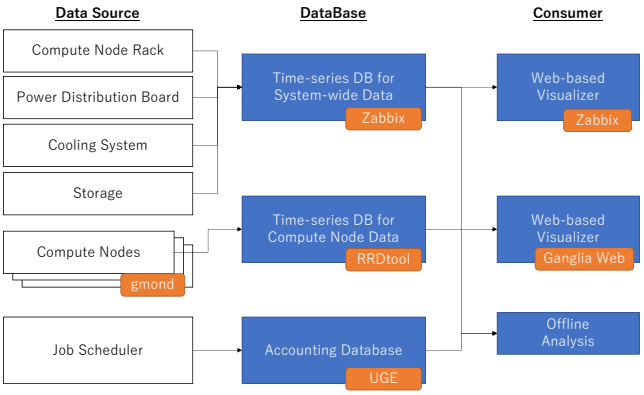

図 1 TSUBAME3 におけるジョブ情報蓄積の構成

TSUBAME における各種センサー情報は，ノード単位の情報を Ganglia， システム全体にかかる情報を Zabbix で取得し蓄積している．ノードのモニタリングにおいては，Ganglia のモニタリングデーモンである gmond を各ノードに常駐させ，load average, CPU 使用量, GPU 使用量, メモリ使用量, GPU メモリ使用量, ノード消費電力, GPU 温度, GPU 消費電力を 5 分毎に取得し，モニタリングサーバ上のデータベースに RRD 形式で格納している．GPU に関する指標はデバイス毎にデータを取得しているが，CPU に関する指標は 2 ソケットの全コアを合わせた合計値で取得し，ノード内にまたがる複数のジョブに関するデータを適切に按分することはできていない．システム全体のモニタリングを行う Zabbix では，ジョブスケジューラの状態 (実行中および待ち状態のジョブ数やノード数など)，ス

トレージの状態や各ラックおよび冷却設備の消費電力情報などを時系列で記録し，SQL データベースに保存している．Ganglia ではそれぞれの計測値をノード単位で管理していることに対し，これらの計測値はノード境界とは異なる単位で分割されているため，ノード毎の情報と合わせた，同一のデータベース環境で管理することが難しく，分離したシステムとして設計した．Ganglia で利用されている Round Robin Database(RRD) 形式では，古いデータについては解像度を落として保管する設計となっているが，TSUBAME では 5 分解像度のデータを 1 年以上保持する設定として，年度境界のタイミングでデータベースをバックアップすることで，ログ取得サーバのストレージ容量と計測点数のトレードオフを克服している．

2.3.2 産総研 ABCI での事例

産総研が運用する ABCI [9,10] では，運用開始当初からジョブ情報，及び各種センサー情報の蓄積を行っている．ジョブ情報は単一の関係データベースに蓄積している．一方，センサー情報は，複数のデータベース，ファイルに分散して蓄積している．その理由は 2 つある．1 つ目は，電源・冷却設備と計算機を導入したベンダーが異なり，それぞれのセンサー情報を別のデータベースに蓄積していたことである．2 つ目は，計算機のセンサー情報は当初はシステム監視を担っていた Zabbix に蓄積していたが，長期間大容量保存を想定しないハードウェア構成をしていたため，ストレージ容量超過が発生したためである．その結果，センサー情報は Prometheus [14] に集約すべく，年度単位で段階的に構成変更を行ってきた．

図 2 に ABCI におけるジョブ情報・センサー情報蓄積システムの現在の構成を示す．各種データの蓄積・アクセス手段は以下の通りである．共有ファイルシステム，及び Prometheus はアクセス制御をしており，許可されたアカウントのみデータへのアクセス可能である．

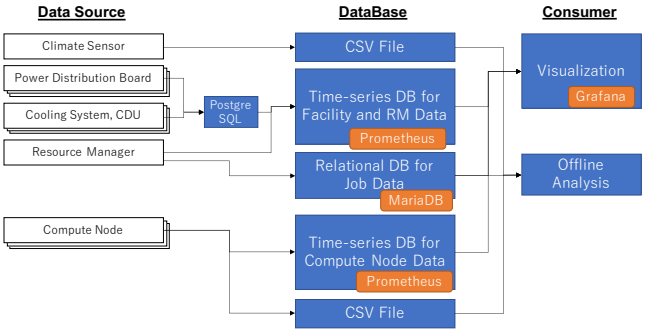

図 2 ABCI におけるジョブ情報蓄積の構成

- ジョブ情報は MariaDB に蓄積している．ジョブ管理サーバからのみ当該 DB にアクセスできる．
- 気象センサーのデータは日単位の CSV ファイルとして共有ファイルシステムに蓄積している．
- 電源・冷却設備のデータは設備導入時に PostgreSQL のデータベースサーバに蓄積する仕様としていたが，孤立したネットワークにあること，及び，他のセンサー情報とアクセス手段を共通化するために，Prometheus に集約している．
- ノードの IPMI から取得されるデータ（ノード電力，温度，ファン回転数など）は，ネットワーク接続の問題で Prometheus を稼働させているサーバでは直に取得することができないため，日単位の CSV ファイルとして共有ファイルシステムに蓄積している．
- IPMI 以外のノードのデータは Prometheus の Node Exporter をノード上で稼働し，Prometheus のサーバに集約している．

集約したデータの一部は Grafana にてリアルタイムに可視化し，利用者が ABCI の混雑状況などの確認に利用している．また，運用チームによるオフラインでの分析用に，Prometheus の HTTP API を用いた各種センサーの生データを CSV 形式で取得するコマンドラインツールを用意している．

2.3.3 上記事例以外のツール群

NVIDIA 社が提供している Data Center GPU Manager(DCGM) [15] は，クラスタ上の NVIDIA GPU および NVSwitch に特化した管理ツールであり，GPU の利用率や温度，ファン速度などの情報を収集する機能を持つ．また，計測値をエクスポートする API が整備されており，dcgm-exporter を利用することで，Prometheus などの他のモニタリングツールに計測情報を取り込むことが出来るように設計されている．上記 2 サイトの事例においては，今のところ利用されていないが，DCGM を用いているサイトにおいてもエクスポート API を活用することにより，比較的容易に計測情報を利活用することが出来ると考えられる．

かつて，2000 年代頃まではスーパーコンピュータは大規模なものでも 1000 ノード程度であり，IBM Tivoli Monitoring [16] や日立製作所 JP/1 Server Conductor [17] などのサーバ管理製品がスーパーコンピュータの管理に使用される場合があった．これらの製品はサーバ運用に必要なハードウェア監視，ソフトウェア管理，ストレージ管理，バックアップなどの機能を提供するパッケージソフトウェア製品群の一部であり，スーパーコンピュータでは一部の必要部分のみを有効化して使用していた．これらの製品は，自社製品のみで管理業務が完結するシステム，あるいは特定の外部ツールとの連携のみを想定している．そのため，その連携方法は独自フォーマット，独自プロトコルを使用するものとなっており，本報告で目指すような共通的なデータフォーマットを規定するものではない．2021 年現在，これらの製品は仮想環境への対応などを強化しており，1 万ノードを超えるようになったスーパーコンピュー

タ向けには進化していない．今後も商用の汎用サーバ管理ソフトウェアをスーパーコンピュータの管理に活用する機会は減少傾向であると考えられるため，本報告ではこのような商用ソフトウェアはデータソースとしても保管手段としても想定しない．

これまでの事例で見てきたように，データセンターにおけるモニタリングツールの選択は，その計測値の性質に応じて複数のツールが使い分けられている現状があり，モニタリングデータの利活用だけを目的に，特定のツールの利用に統一することは現実的ではない．また，これらのモニタリングツール群は基本的に時系列データを蓄積する設計となっており，ジョブスケジューラによるジョブ境界などを意識してデータを分割するなどの機能はない．そこで，本報告では次節に定義する標準ジョブ履歴スキーマによって定義される計測項目を，既存のモニタリングツール類およびジョブスケジューラから得られるデータを加工することによって取得することを提案する．この方式を取ることによって，必要なデータが記録さえされていれば，スケジューラやモニタリングツールに関わらず統一した形式のジョブ履歴データを，追加のモニタリングツールなどのシステム負荷をかけることなく取得することができる．

本節で述べたものを含む各種モニタリングツールの階層関係を**図 3** に示す．本報告の提案する標準ジョブ履歴

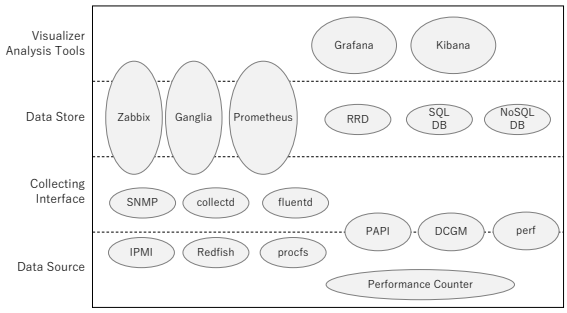

図 3 モニタリングツールと階層関係

スキーマは，本図における Analysys Tools の階層に位置し，下位層の各種ツールの出力をまとめて，共通のインタフェースを提供するものとなる．

3. 標準ジョブ履歴スキーマの提案

3.1 ジョブ履歴スキーマに求められる要件

電力制約などを意識したスケジューリングの実現のためには，従来の SWF，GWF 等のジョブ履歴フォーマットに加えて，以下のような事項を保持できることが望ましい．

- ジョブの具体的な配置に必要な情報
  - ジョブの要求ノード数
  - ノード分割の際の必要サイズ
- センサー情報とジョブを紐づけるための情報
  - ジョブの実行ノードリスト
  - 割り当てられた CPU, GPU の ID (ノード分割を行う場合)
  - センサー情報
- ジョブの依存関係やアレイジョブであるという情報

3.2 時系列データのデータ量とデータ解像度

ジョブ履歴のうち，投入時刻などのジョブ毎に 1 つの値しか持たない指標に較べてノード毎やプロセッサ毎に得られる指標や時系列データとして得られる指標 (主にセンサー情報) はデータ量が膨大となり，無制限に取得するとストレージ容量および解析時の I/O 時間や計算量を圧迫するのみである．一方で，解析の目的によってはその空間方向もしくは時間方向の解像度をできるだけ高くしたいという需要があるところである．

計測点 1 点あたりのデータ保存に必要なストレージ容量を考える．1 点あたり n バイトのデータを，ノード当たり k 点，m ノード分のデータを t_s 秒間隔で取得し，t 秒間蓄積するのに必要なストレージ容量は $kmnt/t_s$ バイトであり，$n = 4$，$k = 50$，$m = 100$，$t_s = 60$，$t = 86400$ の場合，28.8MB となる．これはジョブ 1 つ分の計測データとしては過大であり，このようにノード数などのジョブあたり 1 点しか値を持たないデータと同様に時系列データを扱うことは現実的ではなく，必要性に応じて時系列データを提供できる仕組みを用意しつつも，ジョブ履歴データ本体にはすべての時系列データを直接は含まない仕組みが必要となる．

具体的に要求されるデータの解像度の一例として，Green500 ランキングにおける電力測定基準として用いられている EEHPC Power Measurement Methodology [18] が参考になると思われる．計測の要求レベル別に 3 レベルの計測ポリシーが定められており，レベルごとに要求基準が異なるが，レベル 1〜2 において，電力計で積算電力を推定する場合には，毎秒 1 サンプル以上の粒度で等間隔の計測が要求される．レベル 3 においては，5kHz 以上のサンプリングレートを持つ電力量計を用いて電力量を直接計測することが求められている．ただし，Green500 の計測は，あくまでもシステム全体 (もしくはその一部分) の合計の電力量を 1 か所で計測するものであり，ノードもしくはプロセッサ事の指標を計測し，記録し続けるジョブ履歴における時系列データでは現実的な解像度は異なる．

いずれにせよ，センサー情報からなる時系列データについては，ジョブ履歴データの利用目的によってその要否や必要な解像度が異なるため，ジョブ履歴本体のデータとは別に必要な解像度でデータを取得できるような API が必要になると考えられる．

3.3 標準ジョブ履歴スキーマ

ジョブの履歴情報は，次節以降に示す項目からなるジョブの履歴情報のリストからなる JSON 形式のデータとする．

履歴ファイルに含まれるデータ範囲の指定方法として，開始時刻や終了時刻の指定やジョブ ID の指定などが考えられるが，それらの API については本報告では検討の対象とはしない．ただし，次節に述べる timeline 型データの取得に HTTP(S) プロトコルを活用する点も含めて，何らかの Web インタフェースを作成する形を念頭に置いている．

3.3.1 データ型

ジョブ履歴の各項目については，以下のいずれかの型もしくは JSON 構造体を持つ．なお，型名の代わりに job_id などの項目名を記載することで，その項目の値を持つ要素への参照を表すこととする．

- *number*
- *string*
- *time*
  時刻情報は *YYYYMMDDThhmmss+TZ* で示される ISO 8601 基本形式で記述することとし，1 つのログデータ内では同一のタイムゾーンを用いることとする．この条件下では，時刻情報の辞書順並び替えが時間方向の並び替えと一致する．
- *[datatype]*
  datatype で示されるデータ型のリストを示す．
- *timeline*
  3.2 節における議論のとおり，時系列データについては計測ツールにおけるオリジナルフォーマットでの保存としつつ，利用時に必要に応じて解像度を指定しつつ別途ダウンロードできる URI を用いるものとする．記載された URI の末尾に`?option=value` 形式でオプションを記述することで，出力を調整できる場合があり，`?freq=5min` などの形で利用できることを想定している．
  また，先述の Green500 の例のように，時系列データ本体ではなく，その集計値である合計もしくは最大値のみが必要となる場合が多いことから，*timeline* 型の項目については，項目名の末尾に`_max` および`_avg` を付して，別途巨大な時系列データをダウンロードせずとも，その集約値にアクセスできるようにする．

3.3.2 記録項目

以下の項目を標準の項目名として定義する．このスキーマでジョブの履歴情報を網羅しているわけではなく，必要に応じて同様の命名規則で項目を拡張することと，必要に応じて拡張を標準に取り込むことを念頭に置いて設計されている．**橙色**で示されている要素は必須要素で，それ以外はモニタリング環境の有無や該当性に応じて省略されるオプション要素である．

なお，それぞれの項目の単位については原則的に項目名の接尾辞として含めることとし，数値の意味について解釈の齟齬がないように留意している．

- **jobid**: *string*
  いわゆるアレイジョブの各インスタンスや，ジョブの自動再実行に関するデータについては，適切に jobid をエンコードすることで，別のジョブとして区別する
- queue_time: *time*
  ジョブの投入時刻
- **req_walltime_sec**: *number*
- **resource_req**
  - **num_host**: *number*
  - num_cpu_perhost: *number*
  - mem_perhost_bytes: *number*
  - frequency_policy: *string*
  - accelerator
    - type: *string*
    - num_perhost: *number*
    - mem_perhost_bytes: *number*
    - frequency_policy: *string*
- user_priority: *number* (0〜100)
  大きいものほど高優先のジョブを示すよう適宜変換するものとする．
- job type: *string*
  batch/interactive および array か否かを示す文字列を記載する．
- dependency: *[*jobid*]*
- **start_time**: *time*
- **end_time**: *time*
- exit_status: *number*
  0 がスクリプトの終了，その他は異常終了 (時間・資源超過等)
- **per_host**: *[ ]*
  - **node_id**: *string*
  - start_time: *time*
  - end_time: *time*
    これらはノード毎の実行時間がジョブ全体の実行時間と異なる場合に記載する．
  - used_cpu_percent: *timeline*
  - used_mem_bytes: *timeline*
  - used_vmem_bytes: *timeline*
  - accelerator: *[ ]*
    - utilization_percent: *timeline*
    - usedmem_bytes: *timeline*
  - frequency_hz: *timeline*
  - energy_joule: *timeline*
  - power_watt: *timeline*
  - storage
    - utilization_bytes: *timeline*

    * access_bytes: *timeline*
  - communication_bytes: *timeline*
  - io_bytes: *timeline*
- user_identifier: *string*
  仮名化処理を行うことも想定するが，1 つのログファイル内ではそれぞれのユーザが弁別できることが望ましい．
- accounting_group: *string*
- maxpower_watt: *number*
  電力に関しては，ノード毎の最大値の総和と，ノード毎の総和の最大値が意味するところは異なり，後者の指標が必要となることがあるため，per_host の項とは別に定義している．

4. 実装イメージ

図 4 に本報告で提案するシステムの実装イメージを示す．

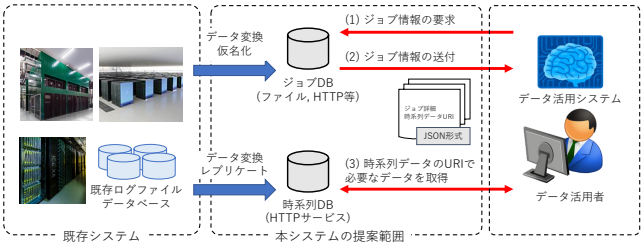

図 4 システムイメージ

本システムでは，既存のモニタリングシステム等が持つデータベースからジョブや時系列データを変換し，保存する．モニタリングシステムのデータベースを直接参照する形での実装も可能ではあるが，データ公開時に求められるユーザ情報の仮名化や，*timeline* 型で示される時系列情報の集約操作などが必要となるため，ジョブ単位のデータについてはスケジューラログの直接参照ではなく，レプリカを作る形で実装することを想定している．

本システムのユーザは，時間情報やジョブ ID などの，対象範囲を指定してシステムにジョブ履歴を要求する．システムでは，3.3.2 節に示されるスキーマに従って，該当するジョブ履歴情報を JSON 形式で作成し，ユーザに提示する．この際に，*timeline* 型で示されるデータについては，事前計算した集約値情報と，取得項目や開始終了時刻をエンコードした URI を返すにとどめ，時系列 DB へのアクセスはこの段階では行わない．ユーザは得られたジョブ履歴情報において，時系列データを必要とする場合にのみ，時系列データ取得用 URI にアクセスし，本システムが要求された範囲の時系列情報を出力する．

5. 活用イメージ

本研究で構築しようとしている，多数のジョブと，それらと 1 対 N で紐づけられた時系列センサー情報からなるワークロードデータベースの標準的スキーマが確立され普及すると，ジョブの属性情報に加え，関連するセンサー情報を用いた，ジョブ特性のモデリングやスケジューリング研究の普及，および，そこで生成・提案されたモデル・アルゴリズムの改善促進が期待できる．

ジョブのモデリングに関する既存研究として，次のような研究がある．京コンピュータのジョブ実行ログ，及びジョブの電力データを用いて，Yamamoto らはジョブのアプリケーション単位での分類予測手法を提案し [19]，鈴木らは将来のシステム全体の消費電力量を予測する手法を提案している [20]．Borghesi らは，Eurora スーパーコンピュータのジョブログを用いた，決定木によるジョブ電力予測 [21]．Zhang らは，データセンター内仮想マシン上で稼働している Linux コンテナの電力消費モデルを提案しているが，独自に構築した数ノードの OpenStack クラスタで取得したデータを用いて評価を行っている [22]．スケジューリングに関する研究としては，以下が例に挙げられる．Wallace らは，過去のジョブの電力データをもとに，これから実行するジョブの消費電力を予測し，システム全体の電力消費量を一定値以下におさえる（Power Capping）スケジューリング手法を提案し，Mira スーパーコンピュータのジョブ実行ログを用いて評価している [23]．Borghesi らは，機械学習による電力予測と制約プログラミングを組み合わせた Power Capping を行うスケジューリング手法を提案し，Eurora スーパーコンピュータのジョブ実行ログを用いて評価している [24]．Mao らはジョブの時系列情報を用いた，深層強化学習によるスケジューリング手法を提案しているが，人工的に生成したジョブを用いた評価を行っている [25]．

これら研究成果では，特定のシステムのジョブ情報による評価しか行っていないため，汎化性能に関しての評価が十分でない．その理由は，これら研究で必要とするジョブ情報，及びそれに紐づけられたセンサー情報の収集，蓄積，利用が困難であることに起因していると我々は考えている．これらの研究を実施するには十分な内容と量のデータが必要だが，そのためには大勢の利用者により利用されるスーパーコンピュータ・クラウドシステムが必要であり，データ蓄積のための専用のモニタリングシステムも必要である．このため，そもそもデータを所有できる機関が限定される．また，モニタリングシステムもデファクトスタンダードとなるシステムが存在せず，システムごとに個別に実装されていたのが実情であり，それによりデータの互換性が保証されていなかった．本研究ではデータフォーマットの共通化を行うことで，データ利用時の障害を減らすことができると考えている．我々の提案を受け入れたデータセンターが増えることで，上記のような研究で利用できるデータ量も増え，研究が活発になることを期待している．

6. おわりに

本報告では，従来の SWF，GWF 形式に代わり，多数のノードからなるクラスタ型計算機アーキテクチャにおいてセンサー情報を用いた高度なジョブスケジューリングを行うための，標準ジョブ履歴スキーマを提案した．本スキーマでは，アクセラレータの存在やノード分割，時系列データおよびその集約データを扱えるようにしたことで，電力情報などを活用した精緻なスケジューリングに活用できる形式となっている．本スキーマに従って成形されたジョブ履歴を用いた処理を記述することにより，ジョブスケジューラやジョブの解析プログラムを他のデータセンターへ容易に適用できるようになると期待している．

今後の課題としては，以下のような問題が挙げられる．本報告の執筆時点では，スケジューラログおよび各種モニタリングデータをもとに，提案する標準ジョブ履歴スキーマに基づいたデータが生成されるに十分な履歴情報を記録できていることの確認までしかできておらず，具体的なデータ取得機構の構築や，サンプルデータの作成に至らなかった．このような履歴データスキーマの評価のためには，データを無理なく作成できることと，それを用いて具体的な解析等のワークフローが構築できることを示すことが求められるため，今後本提案の具体化を，TSUBAME3，ABCI，スーパーコンピュータ「富岳」で行いたいと考えている．その過程においては，ジョブの履歴情報というユーザ固有の情報を扱うこととなるため，本報告の直接の議論の対象とはしないとしたものの，履歴データの取り扱いやその公開方法，仮名化などの情報保護方法についても議論を進める必要がある．

謝辞 本研究は，JSPS 科研費 (JP19H04121) の助成を受けたものである．

参考文献

[1] Feitelson, D. G., Tsafrir, D. and Krakov, D.: Experience with the parallel workloads archive, *Journal of Parallel and Distributed Computing*, Vol. 74, No. 10, pp. 2967–2982 (2014).

[2] Feitelson, D. G., Tsafrir, D. and Krakov, D.: Parallel Workloads Archive, `https://www.cs.huji.ac.il/labs/parallel/workload/`.

[3] Chapin, S. J., Cirne, W., Feitelson, D. G., Jones, J. P., Leutenegger, S. T., Schwiegelshohn, U., Smith, W. and Talby, D.: Benchmarks and Standards for the Evaluation of Parallel Job Schedulers, *Job Scheduling Strategies for Parallel Processing* (Feitelson, D. G. and Rudolph, L., eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 67–90 (1999).

[4] Iosup, A., Li, H., Dumitrescu, C., Wolters, L. and Epema, D.: The Grid Workloads Archive, `http://gwa.ewi.tudelft.nl/`.

[5] The Workshop on Jobs Scheduling Strategies for Parallel Processing: JSSPP Workloads Archive, `http://jsspp.org/workload/`.

[6] Klusáček, D., Tóth, v. and Podolníková, G.: Complex Job Scheduling Simulations with Alea 4, *Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques*, SIMUTOOLS'16, Brussels, BEL, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p. 124–129 (2016).

[7] CESNET: Alea Simulator, `https://www.metacentrum.cz/en/devel/simulator/`.

[8] 松岡 聡, 遠藤敏夫, 額田 彰, 三浦信一, 野村哲弘, 佐藤 仁, 實本英之, Drozd, A.：HPC とビッグデータ・AI を融合するグリーン・クラウドスパコン TSUBAME3.0 の概要，情報処理学会研究報告，Vol. 2017-HPC-160, No. 29, pp. 1–6 (2017).

[9] 小川宏高，松岡 聡，佐藤 仁，高野了成，滝澤真一朗，谷村勇輔，三浦信一，関口智嗣：世界最大規模のオープン AI インフラストラクチャ AI 橋渡しクラウド（ABCI）の概要，第 165 回ハイパフォーマンスコンピューティング研究会 (2018).

[10] 滝澤真一朗，坂部昌久，谷村勇輔，小川宏高：ABCI 上でのジョブ実行履歴の分析による深層学習計算の傾向把握，第 176 回ハイパフォーマンスコンピューティング研究会 (2020).

[11] DMTF: Redfish, `https://www.dmtf.org/standards/redfish`.

[12] Hartmann, R., organization, K., Brazil, B. and Skillington, R.: The OpenMetrics project, `https://openmetrics.io/`.

[13] Altair Engineering, Inc.: Altair Grid Engine, `https://www.altair.com/grid-engine/`.

[14] Authors, P.: `https://prometheus.io/`.

[15] NVIDIA Corporation: NVIDIA DCGM, `https://developer.nvidia.com/dcgm/`.

[16] Gucer, V., Altaf, N., Anderson, E. D., Boldt, D.-G., Choilawala, M., Escobar, I., Godfrey, S. A., Sokal, M. M. and Walker, C.: *IBM Tivoli Monitoring: Implementation and Performance Optimization for Large Scale Environments*, IBM Red Books (2008).

[17] 日立製作所：JP1/Server Conductor, `http://www.hitachi.co.jp/Prod/comp/soft1/serverconductor/`.

[18] EEHPC WG: Energy Efficient High Performance Computing Power Measurement Methodology version 2.0 RC 1.0, `https://www.top500.org/static/media/uploads/methodology-2.0rc1.pdf`.

[19] Yamamoto, K., Tsujita, Y. and Uno, A.: Classifying Jobs and Predicting Applications in HPC Systems, *High Performance Computing*, Springer International Publishing, pp. 81–99 (2018).

[20] 鈴木成人，平岡美智子，白石 崇，クリシュパエンジ，山本拓司，福田裕幸，松井秀司，藤崎正英，宇野篤也：高効率ファシリティマネジメントを実現するジョブ電力予測手法の提案，第 172 回ハイパフォーマンスコンピューティング研究会 (2019).

[21] Borghesi, A., Bartolini, A., Lombardi, M., Milano, M. and Benini, L.: Predictive Modeling for Job Power Consumption in HPC Systems, *High Performance Computing* (Kunkel, J. M., Balaji, P. and Dongarra, J., eds.), Cham, Springer International Publishing, pp. 181–199 (2016).

[22] Zhang, X., Shen, Z., Xia, B., Liu, Z. and Li, Y.: Estimating Power Consumption of Containers and Virtual Machines in Data Centers, *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 288–293 (online), DOI: 10.1109/CLUSTER49012.2020.00039 (2020).

[23] Wallace, S., Yang, X., Vishwanath, V., Allcock, W. E., Coghlan, S., Papka, M. E. and Lan, Z.: A Data Driven Scheduling Approach for Power Management on HPC Systems, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16, Piscataway, NJ, USA, IEEE Press, pp. 56:1—-56:11 (online), available from ⟨http://dl.acm.org/citation.cfm?id=3014904.3014979⟩ (2016).

[24] Borghesi, A., Bartolini, A., Lombardi, M., Milano, M. and Benini, L.: Scheduling-based power capping in high performance computing systems, *Sustainable Computing: Informatics and Systems*, Vol. 19, pp. 1–13 (online), DOI: https://doi.org/10.1016/j.suscom.2018.05.007 (2018).

[25] Mao, H., Alizadeh, M., Menache, I. and Kandula, S.: Resource Management with Deep Reinforcement Learning, *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, HotNets '16, New York, NY, USA, ACM, pp. 50–56 (online), DOI: 10.1145/3005745.3005750 (2016).