高性能計算機システムにおける研究データ管理のための来歴記録システムの実現に向けて

並木 悠太 [1),2)], 細見 岳生 [1),2)], 田主 英之 [1)], 片岡 直記 [1),2)], 山下 晃弘 [1)], 伊達 進 [1)]

1) 大阪大学 サイバーメディアセンター 高性能計算・データ分析融合基盤協働研究所
2) 日本電気株式会社

yuta.namiki@nec.com

Towards Provenance Recording System for Research Data Management on High Performance Computing System

Yuta Namiki[1),2)], Takeo Hosomi[1),2)], Hideyuki Tanushi[1)], Naoki Kataoka[1),2)], Akihiro Yamashita[1)], Susumu Date[1)]

1) Joint Research Laboratory for Integrated Infrastructure of High Performance Computing and Data Analysis, Cybermedia Center, Osaka University
2) NEC Corporation

概要

近年、研究データ管理の重要性が高まりつつある。とりわけ、大容量・大規模なデータを処理する高性能計算機システムでは、入力データ、計算結果等のシステム上で扱われる研究データをどのように管理するかが課題となる。研究データ管理の目的を研究の透明性の確保、研究データの利活用の促進と鑑みれば、高性能計算機システムで生み出される研究データに対しては、その来歴を記録・管理することが必要であると考えられる。本稿では、そのような視点から高性能計算機システム内で扱われるデータの来歴をどのように記録・管理するかについて検討する。その上で、高性能計算機システムにおける研究データ管理を目的とした来歴記録システムの実現に向け、その機能要件を整理する。その後、我々の開発したプロトタイプについて紹介する。

1 はじめに

近年、研究データ管理の重要性が高まっている。これは研究成果の確証となるデータを保全し、研究過程を明らかにすることによる研究の透明性の向上（研究不正の防止）、また、世界的な知の共有、すなわちデータの利活用の促進による研究の効率化が求められていることが背景にある。国内では、内閣府の統合イノベーション戦略推進会議において、研究機関は公的資金による研究に対してデータポリシの策定を行うこと、および機関リポジトリへの研究データの収載を進めることが方針として示されている [1] 。

こうした中、研究データ管理を支える基盤として、国立情報学研究所が NII 研究データ基盤（NII Research Data Cloud、以下 NII RDC と言う） [2] を開発し、2021 年度から運用を開始している。NII RDC は研究データの保存、公開、検索の機能を提供する。これを利用することで、例えば、研究チーム内で研究データの共有、データ識別子を付与しての一般公開、書誌情報に基づいた研究データの検索等が可能となる。

しかし、研究の透明性確保、研究データの利活用促進を鑑みた場合、必要な情報を収集する機能が提供されていないため、現状の NII RDC の導入・利用だけでは困難であると考える。これは NII RDC が研究データの保存、公開、検索を主目的としていることに起因する。すなわち、NII RDC は研究データ管理に必要とされる、データの収集、保存、公開、検索の機能のうち、後半の 3 機能をサポートするものである。

高性能計算機システムは、シミュレーションや実験の環境として日々活用されデータを生み出している。これらのデータを研究の透明性確保、研究データの利活用促進という研究データ管理の観点から、適正に記

録・管理する方法は確立していない。高性能計算機システムを対象とし、上述した視点での研究データ管理の実現には、少なくとも以下の 2 点の検討が必要であると考える。

- ファイルの来歴の記録：研究データ管理で管理すべき情報は生み出されたデータそのものだけではない。透明性を担保するために、来歴を記録することが必要とされる [3] 。来歴はあるデータを得るための入力データと処理を明らかにしたもので、データの処理過程を再現するための情報である。
- 来歴記録のシステム化：来歴を記録する作業は研究の効率を低下させうる。また、情報は統一された構造（スキーマ）で収集、管理されなければ探し出すことすらできず有効活用ができない。これらの枠組み作りを研究者に個々に要求することは、収集する情報とその品質、構造が不均一になり活用しにくくなる。そのため、自動化・機械化すべきである。

大阪大学サイバーメディアセンターは、全国共同施設として全国の研究者の研究活動を支える高性能計算機システムを提供、運営している。今後、計算資源を提供するだけでなく、そこで生み出された研究データの透明性の担保、活用を促進する環境を整えることもシステムの提供者としての責務であると考える。

本稿では、研究の透明性の確保、研究データの利活用の促進の視点から、高性能計算機システム内で扱われるデータの来歴をどのように記録・管理するかについて検討する。その上で、高性能計算機システムにおける研究データ管理を目的とした来歴記録システムの実現に向け、その機能要件を整理する。その後、我々の開発したプロトタイプを紹介する。

2 来歴記録システム

2.1 来歴

高性能計算機システムにおける研究データ管理を考えたとき、その目的を踏まえると生み出されるデータの来歴を記録する仕組みがあるべきと考える。研究データ管理の目的は次の 2 点にあった。

- 研究の透明性を確保すること。研究の過程で発生したデータを日時などのメタデータとともに逐次記録し、データの捏造や改竄 [4] といった不正が行われていないことを示す。また、第三者が研究を再現可能なものにする。
- 研究データの利活用を促進すること。データの信頼性を高め、他の研究者がデータを利用しやすくする。データの共有が進むことで研究の加速、効率化が期待できる。

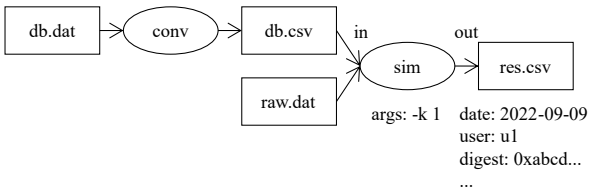

図 1　来歴の例

来歴は前述したとおりデータに対しそれを出力した処理と入力データを明らかにしたものである。ある処理が出力したデータは別の処理の入力データとして処理が連なりうる。図 1 に来歴の例を示す。図中の長方形はファイル、楕円形はプロセス（プログラムによるファイルの処理）、また矢印はデータの流れを示す。この例において、res.csv は sim という処理に db.csv と raw.dat を入力として与えることで得られたことが示されている。換言すれば db.csv と raw.dat を入力とし、処理 sim により res.csv が出力されている。ここで入力ファイルのひとつである db.csv は別の処理 conv により出力されたものであるように、一般に入力、処理、出力の関係が連鎖する。また、ファイルの作成日や作成者などのメタデータも合わせて記録される。

来歴は上に挙げた研究データ管理の目的に次のように貢献する。来歴は入力データとその処理の過程を記録したもので、データを得る手順を明らかにし、不正な処理があればそれも記録に残る。これによりデータの透明性と再現性を向上させる [3] 。また、こうした透明性が高められたデータは信頼性が高いものとなり、他研究者から利用されやすいものになる [5] 。さらに処理とデータが紐づくことで、データもしくは処理単体でなく利活用方法の情報が付加されることも、研究データの価値を高め利活用の促進に繋がることが期待できる。

2.2 利用シーン

高性能計算機システムにおいて生み出される研究データの来歴について、想定される利用シーンを図 2 に示す。この図を用いて研究者、共同研究者、論文査読者、外部の他研究者の 4 つの立場から利用シーンを挙げる。

研究者は高性能計算機システムを利用して各種計測

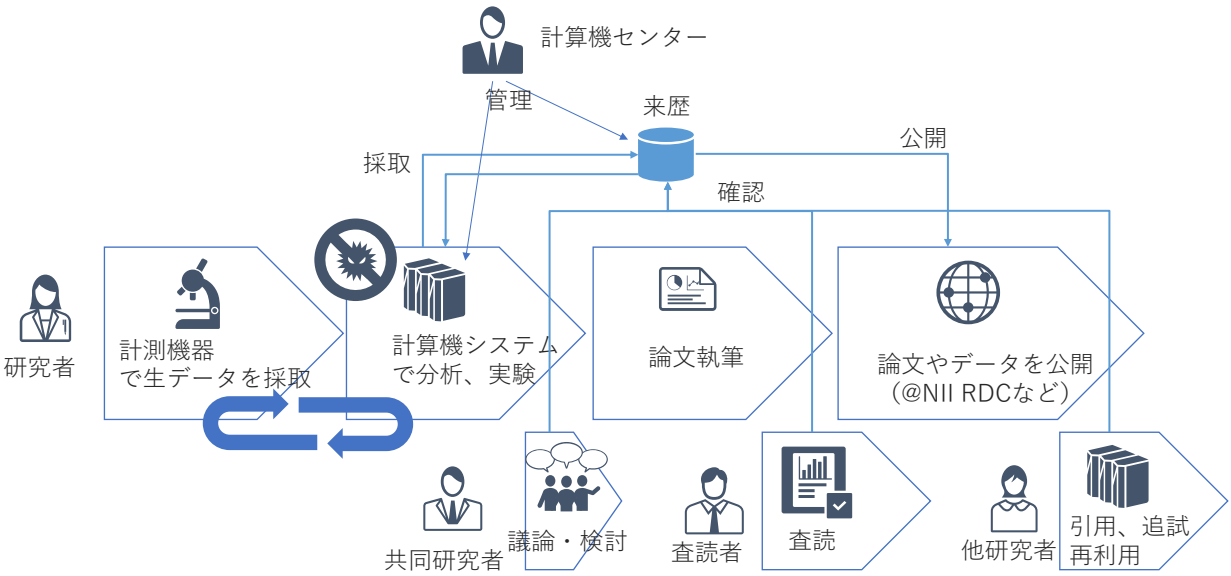

図 2　利用シーン

機器などから得られたデータを基にプログラムを実行してシミュレーションなどを行う。このとき生成されるデータに対し、来歴を記録する。

次に記録された来歴の参照について考える。研究者は記録された来歴を参照できるほか、あるデータが記録された来歴に基づいて生成されたものかを確認できる。これにより論文を執筆する際に最終成果物となるデータの来歴を確認することで、試行錯誤により同じようなデータファイルが大量に存在するような状況であってもそれが生成されたときのパラメーターや過程を確認して意図しないデータの取り違いによる誤りを防ぐことができる。また、研究者は共同研究者に来歴を提示することで、データだけでなく処理の過程も共有しながら研究の議論を行う。さらに、論文の査読者に対しても必要に応じて研究データとともに来歴を提示することで、論文の内容にデータ処理過程との矛盾がないことを示し、研究の透明性、再現性を担保する。外部の他研究者は、元の研究者が公開した研究データを入手し、来歴に示されたものと一致することを確認、すなわちデータが正しい処理の結果であり信頼できるものであることを確認した上で研究に活用する。

2.3　課題

これまでに述べたとおり、来歴は研究データの管理に重要な役割を担う。したがって高性能計算機システムの利用により生み出されるデータに対し、来歴を記録すべきである。

来歴の記録・管理は次の理由によりシステム化すべきと考える。

- 来歴を手動で正確に漏れなく採取、管理することは手間がかかる上に誤りが発生しやすい。高性能計算機システム上でのシミュレーションはパラメーターやプログラム自体を編集し、結果を確認するということを何度も繰り返す試行錯誤のプロセスが含まれる。実験そのものに集中すべき状況で、1 回プログラムを実行するたびに来歴に必要な情報、すなわちパラメーターを含む入力データ、プログラムのソースコード、結果として得られた出力データの組み合わせ、さらにメタデータを記録していくことは、繰り返しの多くが直接的に成果に結びつく結果とならない可能性の高い研究フェーズでは研究者の負担が大きく、研究の効率を低下させる。また、同じファイルを何度も更新する状況では、組み合わせる版を誤る可能性が高い。
- 情報は統一された構造（スキーマ）で収集、管理されなければ探し出すことすらできず有効活用ができない。機械的に情報を収集することで、形式、表記のゆれを少なくすることができる。
- 来歴の記録は研究者本人に依存した自己申告による仕組みでないことが望ましい。これは第三者が記録することにより来歴の信頼性が高まるためである。

ここまでに挙げた課題を以下に整理する。

1. 研究者への負担が小さな来歴とメタデータを記録する仕組みを実現する。

2. 上記をシステム化した、研究者本人に依存しない仕組みを提供する。
3. 記録した情報を基に、データの取り違え、捏造、改竄を検知、防止できる仕組みを実現する。

本稿では、これらの課題を解決する来歴記録システムを実現することに取り組む。

3 設計

3.1 要件

利用シーンを踏まえると、来歴記録システムは 2.3 節で示した課題を実現するために、以下の要件を満たす必要がある。

- 課題 1 への対応
  - 1-1 来歴を記録する：高性能計算機システム上で利用者により生み出された各ファイルについて、来歴を記録する。
  - 1-2 メタデータを記録する：来歴と同様、ファイルの作成者や作成日などのメタデータを記録する。
  - 1-3 ワークロードマネージャー（例：Slurm）や並列処理（例：MPI）環境下で上記の機能が動作する：高性能計算機システムでは、複数の計算ノードを複数の利用者のジョブが競合しないようにしながら効率良く実行させるために一般的にこれらの仕組みが用いられる。
  - 1-4 プログラムの性能への影響が小さい：本稿で対象とする高性能計算機システムは高性能な計算機資源を全学に提供することが第一の目的である。
  - 1-5 プログラムの実行方法など操作への影響が小さい：利用者に新しい操作をすることは負担になるため。
- 課題 2 への対応
  - 2-1 記録はシステムが行う：記録の信頼性を高めるため、記録はその処理を行なった者の申告によるものではなくシステムが自動的に行う。
  - 2-2 記録を保全する：記録はその処理を行なった者やその他の者による情報の捏造や改竄を防ぐため、記録は変更できない。
- 課題 3 への対応
  - 3-1 ファイルやプログラムを与え、その来歴を確認できる。
  - 3-2 来歴およびそこに記録されたファイルとプログラムが捏造、改竄されていないことを確認できる。

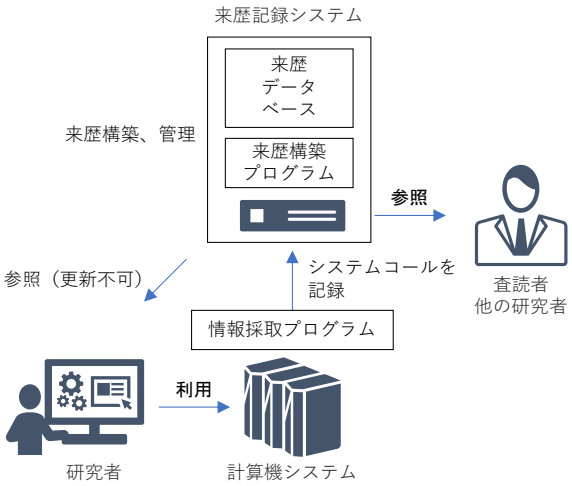

図 3 システム構成

3.2 構成

前項の要件を踏まえ、開発中の来歴記録システムの構成を説明する。システムの構成図を図 3 に示す。本システムは大きく情報採取プログラム、来歴構築プログラム、来歴データベースから構成される。情報採取プログラムは高性能計算機システムを構成する各計算ノードに配置する。このプログラムは計算ノードで生み出されるファイルの来歴の構築に必要な情報を採取し、来歴構築プログラムに渡す。次に来歴構築プログラムは情報採取プログラムが採取した情報をもとに来歴を構築し、来歴データベースに登録する。来歴データベースは来歴を蓄積するとともに、それを参照および検索する機能を提供する。来歴構築プログラムと来歴データベースは高性能計算機システムとは別の計算機で動作することを想定している。

3.3 実現方式

本項では上に述べた構成のもと、3.1 節の各要件の実現方法を説明する。

3.3.1 来歴とメタデータの記録

要件 1-1、1-2、2-1 に対応する。来歴は高性能計算機システムの利用者が実行するプログラム（以下、ユーザープログラムと言う）が発行するファイル操作に関わるシステムコールの記録から構築する。Linux ではプログラムはファイルからデータを読み出す際には OS（カーネル）の `read()` システムコールを呼び出すことで処理を行う。情報採取プログラムはこの呼び出しを検知し、呼び出したプログラム、すなわちユーザープログラムと対象のファイルを記録する。同様

に、ユーザープログラムが書き込んだファイルも対応するシステムコールの呼び出しを検知して記録する。

さらに情報採取プログラムは上記の来歴に関係するファイルについて、ファイルシステムから作成者や作成日などのメタデータを得る。加えてファイルに対してハッシュ値を計算する。これは後述するようにファイルの改竄の検証に用いる。

各計算ノードで動作する情報採取プログラムは、情報を採取するとそれを来歴データベース中の一時領域に保存する。来歴構築プログラムは定期的にこの情報から来歴を構築する。来歴の構築は、一時領域にあるシステムコールの呼び出しの記録を基に、あるユーザープログラムが読み込んだファイルをその入力、書き込んだファイルを出力とすることにより行う。加えてファイルの作成者や作成日、ハッシュ値などのメタデータをファイルに紐付け、来歴データベースに記録する。

3.3.2 高性能計算機システム環境への対応

要件 1-3 に対応する。ワークロードマネージャーのジョブの情報はメタデータとして記録する。これによりジョブと来歴を関連付ける。また、MPI などにより複数の計算ノードで並列実行されたプログラムの場合は、各ノードから採取された記録を統合する必要がある。来歴構築プログラムは情報採取プログラムが一時領域に保存した情報から来歴を構築する際に、プログラムの名称と起動時刻を基に各ノードから得た情報の中から同一のユーザープログラムの実行か否かを判断する。同一と判断したものについて、すべての読み込み、書き込みを行なったファイルをそれぞれ入力、出力として来歴上単一のプログラムに統合した来歴を構築する。

3.3.3 高性能計算機システムの利用者への影響の極小化

要件 1-4、1-5 に対応する。上に述べたように、来歴の構築に必要な情報は、カーネルから採取をする。この手法により、ユーザープログラムに情報取得のための変更を要求することなく、また、ユーザープログラムの実装言語 (C 言語や Fortran など) に依存せず、来歴の構築に必要な情報を得ることが可能である。ユーザープログラムの性能への影響については後述する。

3.3.4 記録の保全

要件 2-2 に対応する。来歴管理データベースは高性能計算機システムとは別のシステムとして運用する。また、高性能計算機システムの利用者と同じシステムで動作する情報採取プログラムは管理者権限で動作させるため、管理者（システムの運営を担う計算機センター）以外がその動作を変更することは不可能である。これにより来歴記録システムの動作や管理するデータがその管理者以外により捏造、改竄されることを防ぐ。

3.3.5 来歴の確認

要件 3-1 に対応する。ファイルやプログラム（実行ファイル）を特定する情報が与えられると、来歴管理データベースはそのファイルが記録された来歴を検索し、要求者に返却する。ここで要求者は特に限定せず、ファイルを入手した共同研究者、査読者、外部の他研究者など任意の人物を想定する。来歴の検索は、例えば与えられたファイルのハッシュ値を計算し、来歴データベースからメタデータにその値を持つファイルの記録を検索することにより行う。来歴記録システムは該当する記録を一覧で要求者に返却する。

3.3.6 ファイルの捏造、改竄がされていないことの確認

要件 3-2 に対応する。まず、来歴そのものの捏造、改竄を防ぐため、前述のとおり来歴記録システムは高性能計算機システムの利用者から可能な限り隔離する。

あるファイルについて捏造がないことは、上記の来歴の確認により当該ファイルの来歴が想定された処理を表しているか確認することにより行う。例えば実験プログラムの出力であるべきファイルがエディターからの出力として記録されているならば、捏造の可能性がある。

ファイルおよびプログラムの改竄のないことの確認も同様である。例えば来歴中に実験プログラムで処理することはあっても手動での編集が想定されていないファイルに対し、エディターでの処理が記録されているならば、改竄の可能性がある。また、改竄されていない正しいプログラムで処理したことの確認は、来歴記録システムに正しいことが判明しているプログラム（実行ファイル）を与え、それと来歴に記録されたプログラムが一致するかをハッシュ値を計算して比較すること、あるいは来歴を参照してそのプログラムが想定どおりのソースファイルからビルドされたものであることを確認することで行う。

4 プロトタイプ

これまでに述べた機能を実験的に実装したプロトタイプを開発した。実装に用いた主な技術を表 1 に示す。また、以下では括弧内に 3.3 節の対応する機能を記述する。

情報採取プログラムにおけるシステムコールの呼

表 1　プロトタイプの実装に用いた技術

システムコールの呼び出しの検知	BPF
実装	C（BPF）、Python（その他の部分）
来歴データベース	Apache Atlas [6]

び出しの検知には BPF [7] を利用する（3.3.1）。BPF は Linux カーネルが提供する、システムコールなどカーネル内の機能が呼び出されたときに開発者が実装した処理、すなわち本システムであればシステムコールの対象となったファイルの情報を来歴構築プログラムに渡す処理を呼び出す仕組みで、カーネルの中の動作のモニタリングが可能になる。本機能は Red Hat Enterprise Linux や Ubuntu などの一般的な Linux ディストリビューションで利用可能な状態にあり、カーネルの入れ替えなどは不要である。また、ユーザープログラムの性能面への影響は小さい（3.3.3）。ファイル入出力が多発するストレージのベンチマークツールである fio [8] を使用して BPF によるファイル操作に関わるシステムコールのトレースのオーバーヘッドを確認した結果、2% 程度であることを確認している。これは同様にシステムコールをトレース可能な ptrace や SystemTap として有利な結果であった。

MPI による並列プログラムの来歴の統合（3.3.2）の機能は来歴構築プログラムの中に実装し、高性能計算機システムとは別の来歴記録システムのためのサーバーで動作させる。

来歴データベースには、オープンソースソフトウェアの Apache Atlas を用い、来歴記録システムのためのサーバーで動作させる（3.3.4、3.3.6）。Apache Atlas はデータ資産の来歴およびメタデータを管理するために特化した構造を持つソフトウェアである。データの蓄積のほか、来歴やメタデータを表示するための画面も提供している。来歴構築プログラムがファイルとプロセス、およびその間の関係性やメタデータなどの情報を定められた形式で登録することで後述するようなグラフィカルな出力を得ることができる。

図 4 に構築した来歴を表示する画面を示す。ファイル名やハッシュ値などを与えて検索することで、対象のファイルについてこの結果が得られる（3.3.5）。図中の青がデータ、黄緑がプロセス（処理）を示しており、矢印はデータの流れである。データからプロセスへの矢印はプロセスがファイルからデータを読んだことを、プロセスからデータへの矢印はプロセスがデータを書き出したことを示している。前者がプロセスに

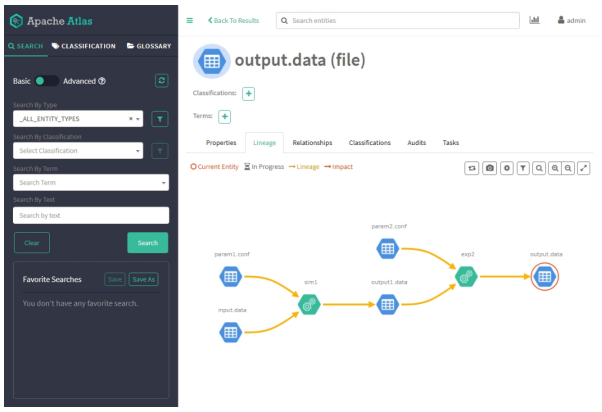

図 4　来歴の表示

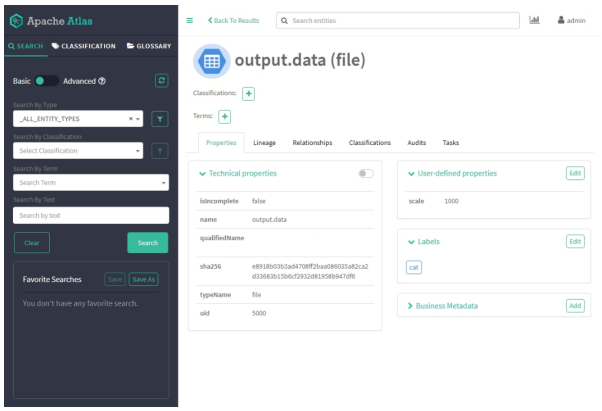

図 5　メタデータの表示

対する入力ファイル、後者が出力ファイルである。

図 5 は収集したファイルやプロセスに紐づくメタデータを表示する画面である。図 4 で表示されているファイルやプロセスから本画面へ遷移することが可能である。ここでは作成者、作成日、ハッシュ値などのメタデータを確認することができる。

5　まとめと今後の課題

本稿では高性能計算機システムにおける研究データ管理を目的とした来歴記録システムの実現に向けた検討を行った。まず、高性能計算機システムにおいて生み出されるデータについて、研究データ管理の目的である研究の透明性確保、研究データの利活用促進を鑑みたとき、来歴を記録・管理することが必要であることを述べた。次に、来歴を記録・管理するための機能

要件を整理した。最後に、これらの要件を実現するために我々が開発したプロトタイプを紹介した。

今後、利用者の意見を取り入れながら、実用性・有用性の高い機能の拡充を進める予定である。また、本システムをすでに運用中の研究データの蓄積、共有を可能にするデータ集約基盤 ONION [9] や GakuNin RDM などの既存の研究データ管理の仕組みと連携させ、研究データの発生から公開までの一連のフローを支援するシステムの在り方の検討を進める。これにより、高性能計算機システムで発生するデータをより簡単、確実に管理し、研究の透明性確保やデータ活用による研究の効率的な推進に貢献する仕組みの実現を目指す。

参考文献

[1] 統合イノベーション戦略推進会議：公的資金による研究データの管理・利活用に関する基本的な考え方. https://www.mext.go.jp/content/20210608-mxt_jyohoka01-000015787_06.pdf.

[2] 国立情報学研究所オープンサイエンス基盤研究センター：NII 研究データ基盤. https://rcos.nii.ac.jp/service/.

[3] National Academies of Sciences, Engineering, and Medicine: *Reproducibility and Replicability in Science*, The National Academies Press, Washington, DC (2019).

[4] 文部科学省：研究活動の不正行為に関する特別委員会報告書. https://www.mext.go.jp/b_menu/shingi/gijyutu/gijyutu12/houkoku/attach/1334660.htm.

[5] Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E. et al.: The FAIR Guiding Principles for scientific data management and stewardship, *Scientific data*, Vol. 3, No. 1, pp. 1–9 (2016).

[6] The Apache Software Foundation: Apache Atlas. https://atlas.apache.org/.

[7] The kernel development community: BPF Documentation. https://docs.kernel.org/bpf/.

[8] Axboe, J.: fio—Flexible I/O tester. https://git.kernel.dk/cgit/fio/.

[9] 伊達　進，寺前勇希，勝浦裕貴，木越信一郎，木戸善之：大阪大学のデータ集約基盤 ONION，大学 ICT 推進協議会 2021 年度年次大会論文集，pp. 130–137 (2021).