RDM/レポジトリにおけるストレージ仮想化レイヤの導入についての検討

理化学研究所　情報統合本部　基盤研究開発部門
データ管理システム開発ユニット
實本　英之*
成島　雅人
平岡　千明

RDM/リポジトリ(ソフト)からのデータ転送の難点

- 文献リポジトリソフトウェアを発展させた考えを持つ
  - 収集・分類整理・長期保存といった図書館的業務機能
  - 論文のプレプリント等の公開場所としての機能

  → 対象を文献からデータ全体に

- 文献->データに当たって発生してくる問題
  - データのキュレーション
    - データ生成者ですらどのデータがデータ管理に値するか判断できていない
  - データの使い方
    - データセットをどのような単位で保存し、どのように切り出して使うのは千差万別
    - どのデータセットがどのような頻度で利用されるかも千差万別
  - データ量の規模感の違い
    - 大規模データの存在・小規模大量データの存在
      - 小規模少数データを主眼としたソフトウェアが元になっている
      - 適切な可視化・アクセスインターフェースがない

分散データストレージ

- 分散データストレージとは
  - 地理的・組織的に分散したストレージインフラを一つのストレージとして見せる機能
    - 複数インフラ間でデータを送受信する機能
    - レプリケーションによるキャッシングや冗長化機能
  - 研究データの共有等で利用している機関がいくつかある
    - Gfarm: HPCI 等
    - iRODS: KEK (今回はこれを例として話を進めていきます)
    - Croudian: 金沢大学等
      - 是非利用感を聞きたい

IRODS:INTEGRATED RULE-ORIENTED DATA SYSTEM

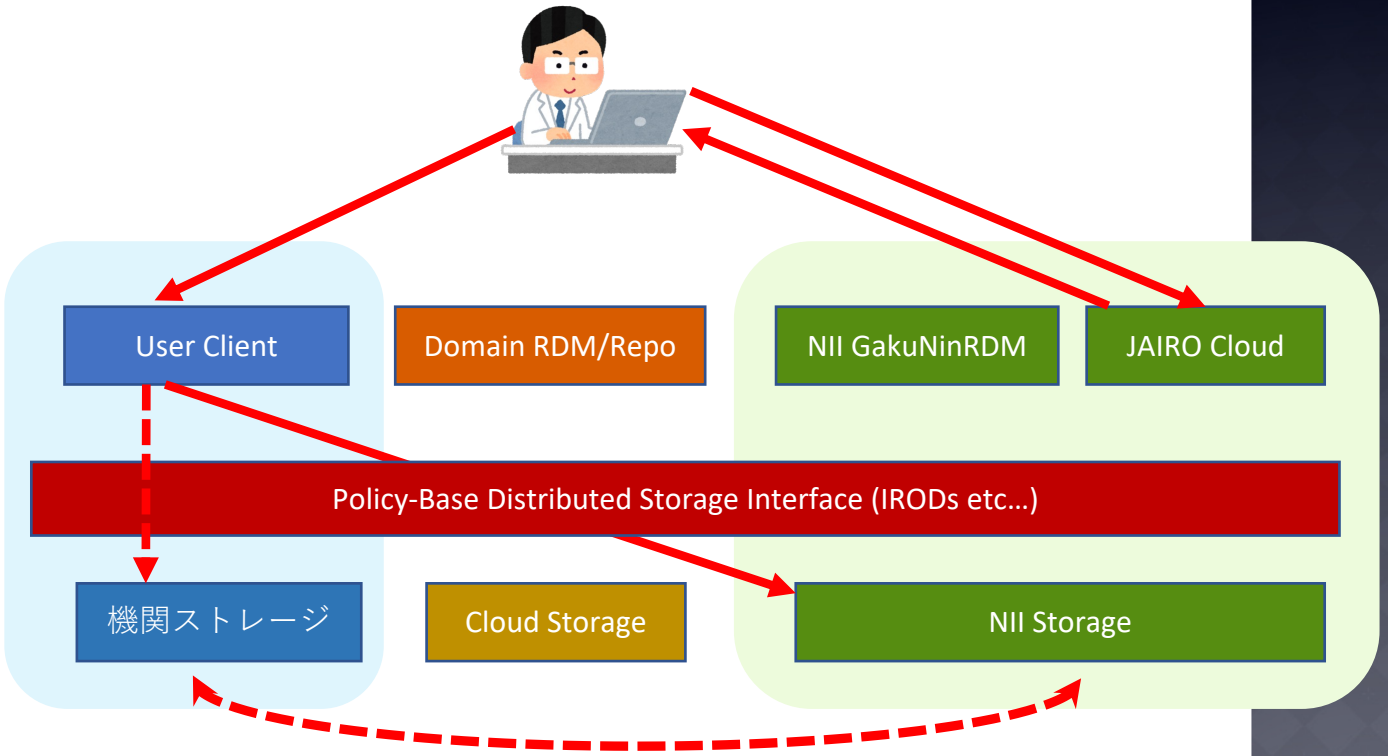
- ルールベースでのデータ管理が可能な分散データシステム
  - オープンソースデータ管理ソフトウェア
    - iRODSコンソーシアムで開発維持が続けられている
      - RENCI(UNC ChapelHil, DUKE, NCSU)を中心に米国立環境健康科学研究所(NIEHS), TACC, DDN, Fujifilm 等
  - 代表的な4機能
    - Data Virtulization
      - データ仮想化により、分散されたストレージを統一された名前空間で扱う
    - Data Discovery
      - メタデータDBを持ち、各種ストレージリソースをメタデータカタログにより検出できる
    - Workflow Automation
      - データの取り扱いをトリガーにしてコールバックサービスを呼び出すことによりワークフローを自動化することができる。
    - Secure Collaboration
      - データ送受信を仲介し、適切なストレージに保存されたデータに適切なコントロールをもってアクセスできる。

RDM/リポジトリに対するIRODS適用シナリオ

- 機関ストレージ等のRDMに接続するストレージをiRODSでまとめることでデータの取り扱いを制御する
  - データ転送・配置に関する問題の解決
    - データ転送をはじめから考慮したシステムの利用による転送高速化
      - スレッド並列転送
    - ファイルサイズ等によるデータ操作方法の変更
      - データが大きい時はRDM経由のダウンロードをせず、iRODSクライアントによる直接ダウンロードを提示する
      - データの保管場所を限られる場合、閲覧だけを認める
    - レプリケーションによる転送速度の向上
      - ファイルのダウンロード時にユーザの近くのリソースからデータを取得する。
      - ユーザの近くのリソースにファイルが存在しない場合には、データのレプリカを作成する
  - データ利用法に合わせた効率的なストレージ利用
    - ポリシーベースバックアップによる凍結データの扱い
      - データ生成者等がわからないこともデータの利用法を監視しながら、適切な扱いを動的に行うことができる
      - 一定期間利用されていないデータを指定したバックアップリソースに移動する等

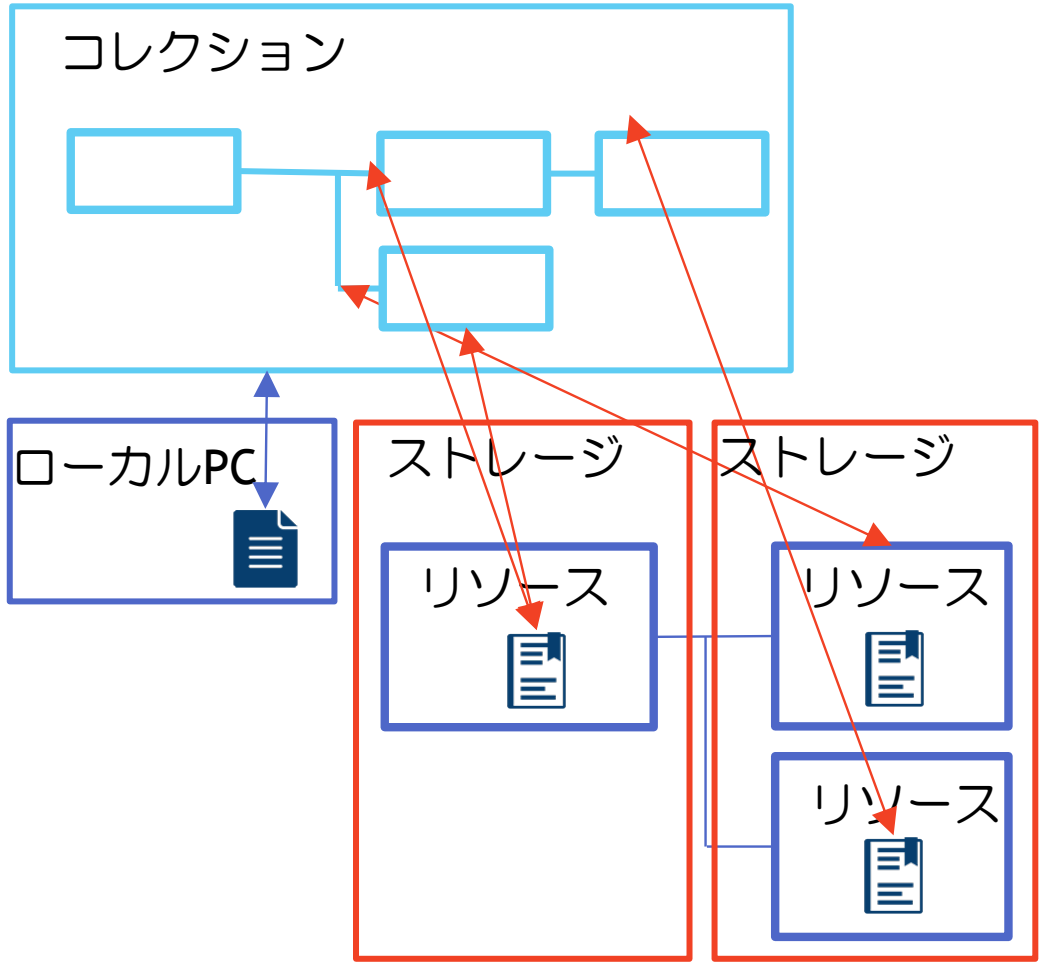
RDMリポジトリに対するIRODS適用シナリオ

- NII ストレージ上の巨大データを機関 A のユーザが反復的に利用する例
  - ユーザがJAIRO Cloud にアクセスし、iRODS での仮想パスを渡される
  - 仮想パスを利用してUser Client 経由でNII ストレージ上データを参照利用する
  - 次回の利用に備え、ユーザから近いストレージにデータをレプリケーションする
    - 次回のアクセスからはiRODSが自動的に機関ストレージのデータを参照するようになる



IRODS の仕組み

- コレクション
  - 全てのデータオブジェクトはコレクションに論理的に結び付けられる
  - コレクションはユーザ毎に作成可能だが、適切な範囲で共有できる
  - コレクション内にさらにサブコレクションを作ることも可能
- 利用者は最寄りのストレージを設定することができる
  - アップロード時特にリソースを指定しない場合は最寄りリソースにアップロードされる。
  - レプリカがある場合は最寄りリソースを優先で取得する



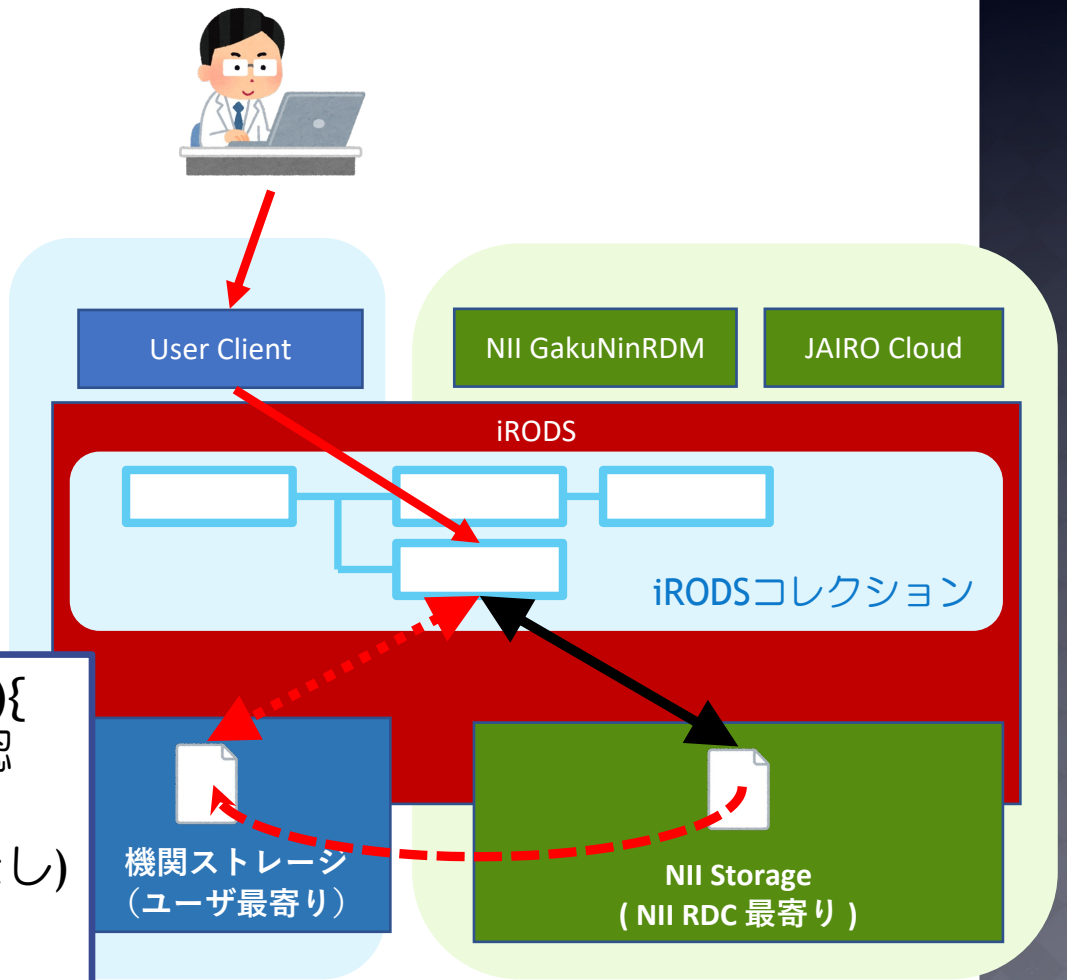
iRODS の仕組み

- Policy Enforcement Points (PEP)
  - データ操作中にポリシーの呼び出しが可能となるトリガーポイント
  - 利用例
    - ユーザの削除に伴うPEP にデータオブジェクトの所有権移行を設定しておき、所有者不明データの発生を防ぐ
    - データのアップロードに伴うPEP でデータ内タグの抽出などを行い、自動的にメタデータを登録する
    - データアップロード／リードのPEP で暗号化／復号化処理を行い、ストレージ上でデータが暗号化して保存されるようにする
  - ルールは iRODS 特有言語／Python／C++ 等で記述できる

ルールエンジンを利用した検証中の実装例

レプリケーションによるデータ転送の隠蔽

- 外部拠点からの読み出し
  - ユーザがNII Storage上データを指すiRODS のコレクションを読み出す
  - ユーザはNII Storage からデータ提供される
  - データ呼び出し回数やレプリカの数などを参照し、iRODSレプリカコマンドを発行する
  - iRODS はレプリカをユーザの最寄りストレージに作成する
  - 次回から高速に読み出せる

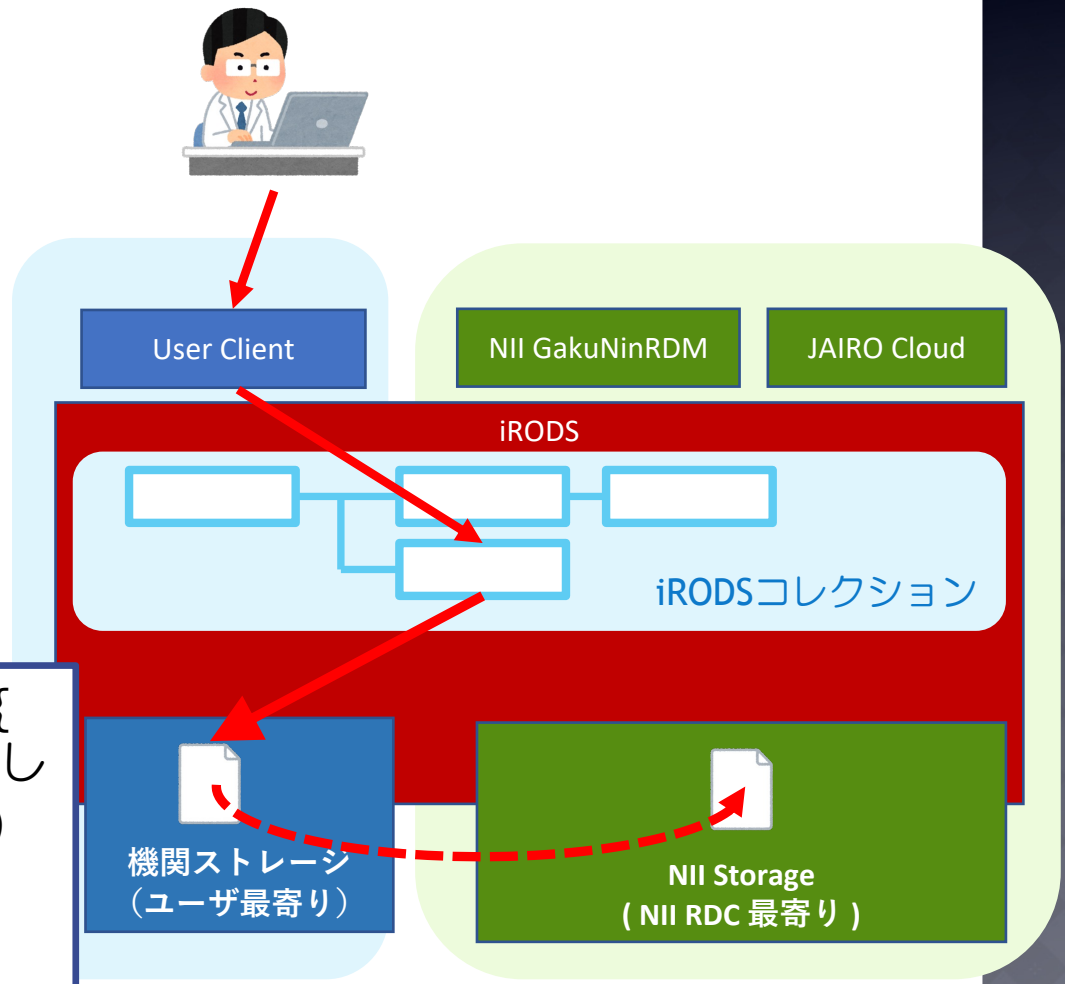



```
データ読み出しPEP(){
    呼び出し回数確認
    if (3回以上
          &レプリカなし)
          irepl 実行
}
```

レプリケーションによるデータ転送の隠蔽

- 外部拠点からの書き込み
  - ユーザが iRODS のコレクションにデータを書き込む
  - 書き込まれたデータは最寄りストレージに保存される
  - データサイズやユーザの意向、レプリカの数などを参照し、iRODSレプリカコマンドを発行する
  - iRODS はレプリカをユーザNII ストレージに作成する
  - NII RDC 内でデータが効率的に連携できる

```
データ書き込みPEP(){
    if (NII Storage になし
            &サイズ確認)
        irepl 実行
}
```



凍結データの蓄積 (VERY COLD DATA)

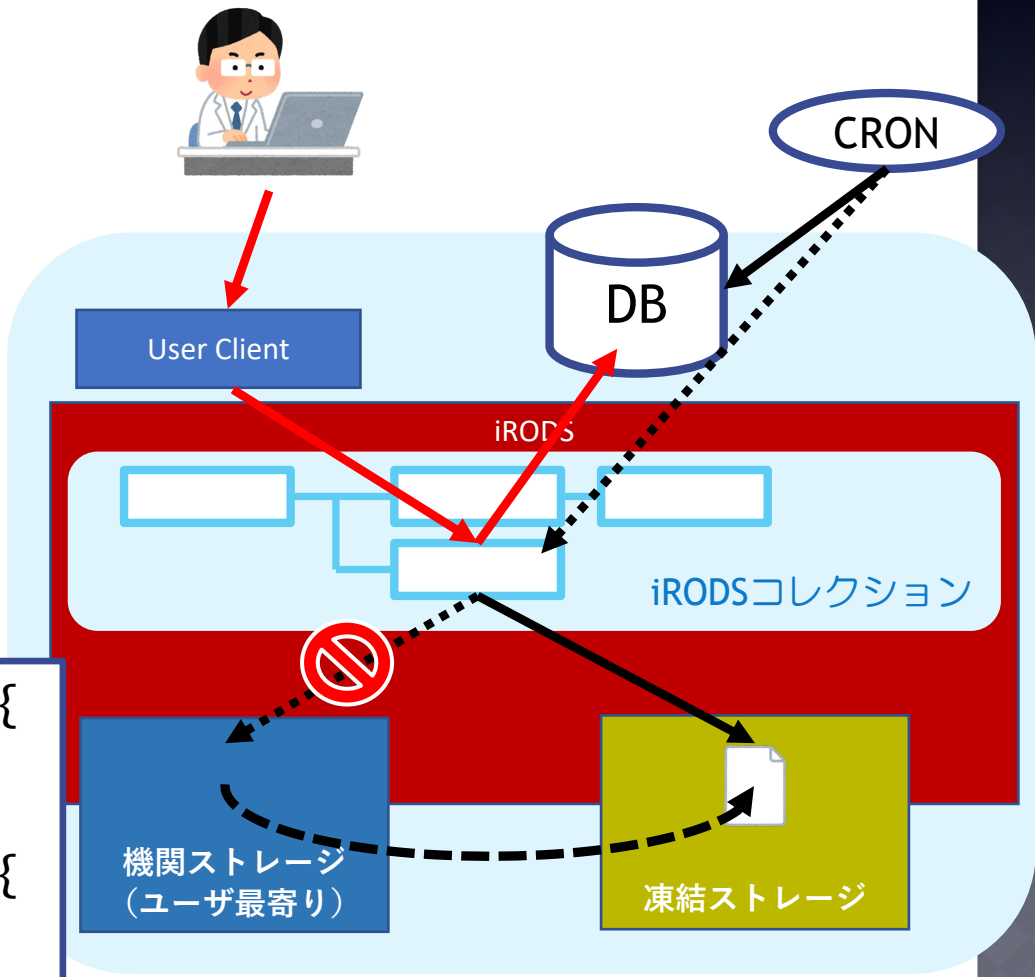
- オープンアクセスとして提出されているがアクセスがほとんどないようなデータに関しては適切な保存媒体に移行すべき
  - テープドライブ
  - 長期保存用クラウドストレージ(AWS Glacier など)
    - 多くの場合データ取り出しに長時間かかる

- iRODSには定期的に全コレクションを対象に実行されるPEPはない
  - 外部DBを用意して管理する。
    - オブジェクト数等の規模感は不明

（余談）テープドライブにはデータのアクセス頻度を見て適切にテープに移す仕組みが大抵あるのですが……

凍結データの蓄積 (VERY COLD DATA)

- 読み書き時
  - ユーザのデータ操作に伴い、作業時間をデータベースに書き込む
- CRON による定期的作業
  - DBを参照し規定時間、操作がないものに関しては保存場所を凍結用ストレージに変更する
  - iRODSのコレクションにアクセスすれば、これまでと同様、データにアクセスはできる（時間はかかる）




```
データ書き込みPEP(){
    日時をDB登録
}
データ読み込みPEP(){
    日時をDB登録
}
```

マウンタ

- ユーザ利便性やスパコン連携を始めとするシステムインテグレーションを考えるとマウンタがあると便利
  - Irods-fs
    - irods コマンドを fuse でファイルシステム化したもの
  - nfsrods
    - NFSv4.1 サーバーを介して iRODS コレクションを公開するもの
    - コンテナによるサービス提供

- 問題点
  - アクセスコントロールの設定が難しい
    - Linux アカウント(httpd ユーザアカウント)と iRODSアカウントの整合性
  - POSIX をフル装備していないかもしれない→調査中
    - httpdアカウントでの手動操作はできるが、NextCloud 等のストレージとして扱おうとすると読み込みのみしかできない (nfsrods)

  →機関ストレージアドオンの新規実装が必要かもしれない

想定される問題

データの正当性保証

- データレプリケーションによる転送時間の隠蔽をするにはデータ取得場所を GakuNinRDM 経由以外に用意する必要がある
  - ユーザクライアントや別RDMの参照ストレージとするなど

→GakuNinRDM 以外がデータを直接触ることになり、
データ正当性保証機能が壊れる

- 解決策として想定できるもの
  - GakuNinRDM-Lite の提供
    - データ操作だけを行えるような何らかの GakuNinRDM と連携したツールを導入する
  - 正当性無保証ディレクトリの導入
    - 必ず無保証ディレクトリにデータを書き込みGakuNinRDM 本体上から各プロジェクトのデータとして反映するようにする
  - GakuNinRDM の iRODS アドオンの作成
    - iRODS のメタデータとやり取りをして GakuNinRDM 上で操作履歴を追えるようにする (データのやり取りに iRODS クライアントをそのまま使える)

機関ストレージ統合の合意

- 各機関のストレージを iRODS として統合するためのルール作りや管理組織の作成
  - 例：HPCI (Gfarm利用)は立ち上げに数年かかっている
- 救いのある点
  - 特定のプロトコルを使えればリソースとして参加可能
    - S3, HPSS, WOS(DDN) 等
    - アドオンを書けば他のプロトコルも？

まとめ

- RDM/レポジトリにおけるストレージ仮想化レイヤの導入についてiRODS を例として紹介
  - 大規模ファイルの取り扱いへの寄与
  - コールドデータの取り扱いへの寄与
  - 計算環境との連携の寄与
  - 問題点の想定

- 今後
  - 現状では GakuNinRDM へのインテグレーションにはパーツが足りない
    - 安定性や性能等の測定を優先し、インテグレーションに進む(コストを支払う)べきかの判断が必要
  - 他の分散ファイルシステムの検証
    - Gfarm は NextCloud と連携済みのため、そのまま GakuNinRDM に組み込める