ハイスループット計算ツール moller の紹介

CCMSハンズオン: moller講習会
2024年10月18日 @物性研A614+Zoom

東京大学物性研究所 附属物質設計評価施設
ソフトウェア開発・高度化チーム

Outline

- スパコンでプログラムを実行する
  - バッチジョブ実行
- ジョブをまとめて実行する: moller
  - moller とは？
  - moller を使う
  - moller をインストールする
- HTP-tools (moller+cif2x) 紹介

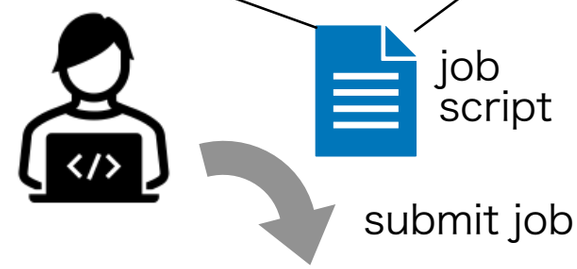
スパコンでプログラムを実行する

- スパコンでプログラムを実行するには:
  - 通常はバッチジョブ形式で実行する
    - ユーザはノード数・実行時間などをリクエスト
      ↓
      ジョブ管理システムがリソースを割り当て
  - リソース量や実行内容をジョブスクリプトに記述
    - シェルスクリプト形式
    - コメント部分にジョブスケジューラへの指示を記載
- ジョブクラス
  - 最大ノード数や実行時間などで区分(システムによる)

```
#!/bin/sh
#SBATCH -p i8cpu
#SBATCH -N 2
#SBATCH -n 8
#SBATCH -c 32
#SBATCH -t 00:10:00

ulimit -s unlimited


srun HPhi -s stan.in
```



スパコンでプログラムを実行する

物性研システムB (ohtaka)に用意されている
ジョブクラス(キュー/パーティション)

(例) F 4 cpu

F	標準　　(24時間まで)
B	低優先度 (12時間まで)
L	長時間　(120時間まで)

cpu	CPUノード
fat	FATノード (大メモリ)

利用可能なノード数

CPUノード

1	1ノード
4	2〜4ノード
16	5〜16ノード
36	17〜36ノード
72	72ノード
144	144ノード

FATノード

2	1〜2ノード

その他に

i8cpu　インタラクティブ・デバッグ (CPUノード)　(30分まで)
i1fat　　同上 (FATノード)

ジョブをまとめて実行する

パラメータサーチなど、条件を変えながら
同じ計算を大量に実行する場合に
スパコンを効率よく使いたい

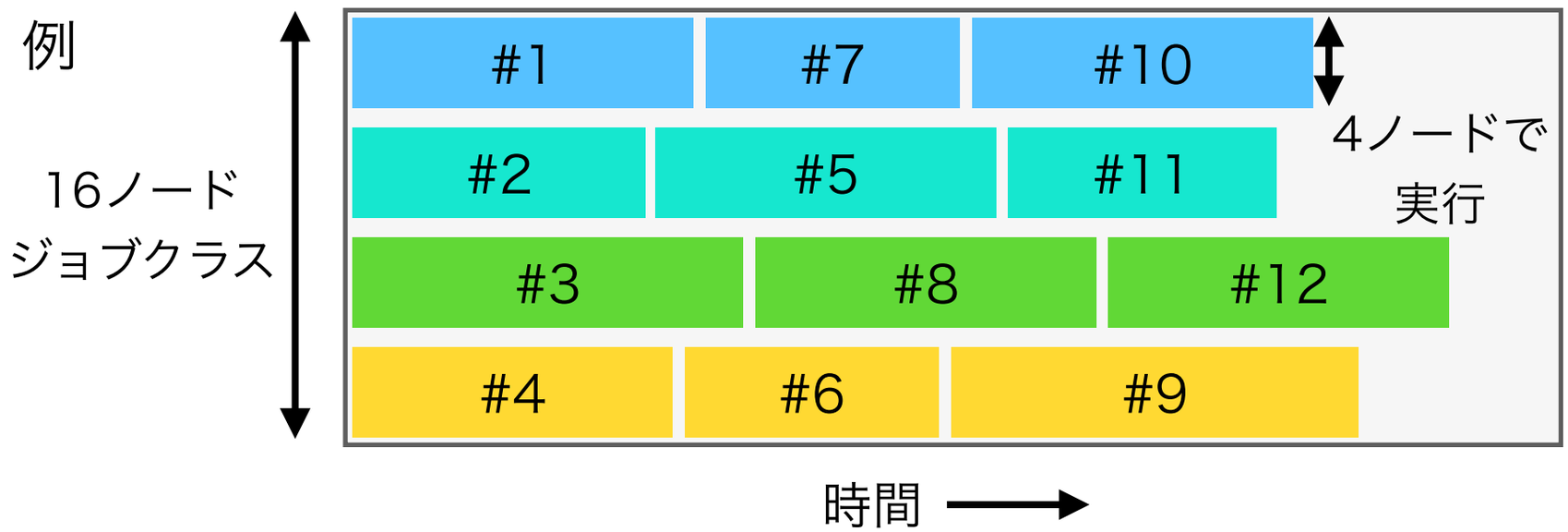
多数のジョブを
1つの大型のジョブクラスで
一括して実行 (バルク実行)

ジョブをまとめて実行する

- 例: 4ノードを使ったMPIプログラムの実行を、12通りの入力データに対して行う

  A. 小型のジョブクラス(4ノード・クラス)で個別のバッチジョブとして実行する

  B. 大型のジョブクラス (例: 16ノード・クラス) のバッチジョブの中にジョブを束ねて実行する (バルク実行)



ジョブをまとめて実行する

- メリット:
  - ノード内の複数コアを効率よく使える
  - 細かいジョブを大量に投入するとジョブスケジューラの負荷・オーバーヘッドが大きい →1つのバッチジョブにまとめて実行
  - 大型のジョブクラスは比較的空いている(こともある)
- デメリット:
  - バルク実行のやり方はプラットフォーム依存
  - →プラットフォーム依存性を隠蔽する汎用ツールを作成

ジョブをまとめて実行する

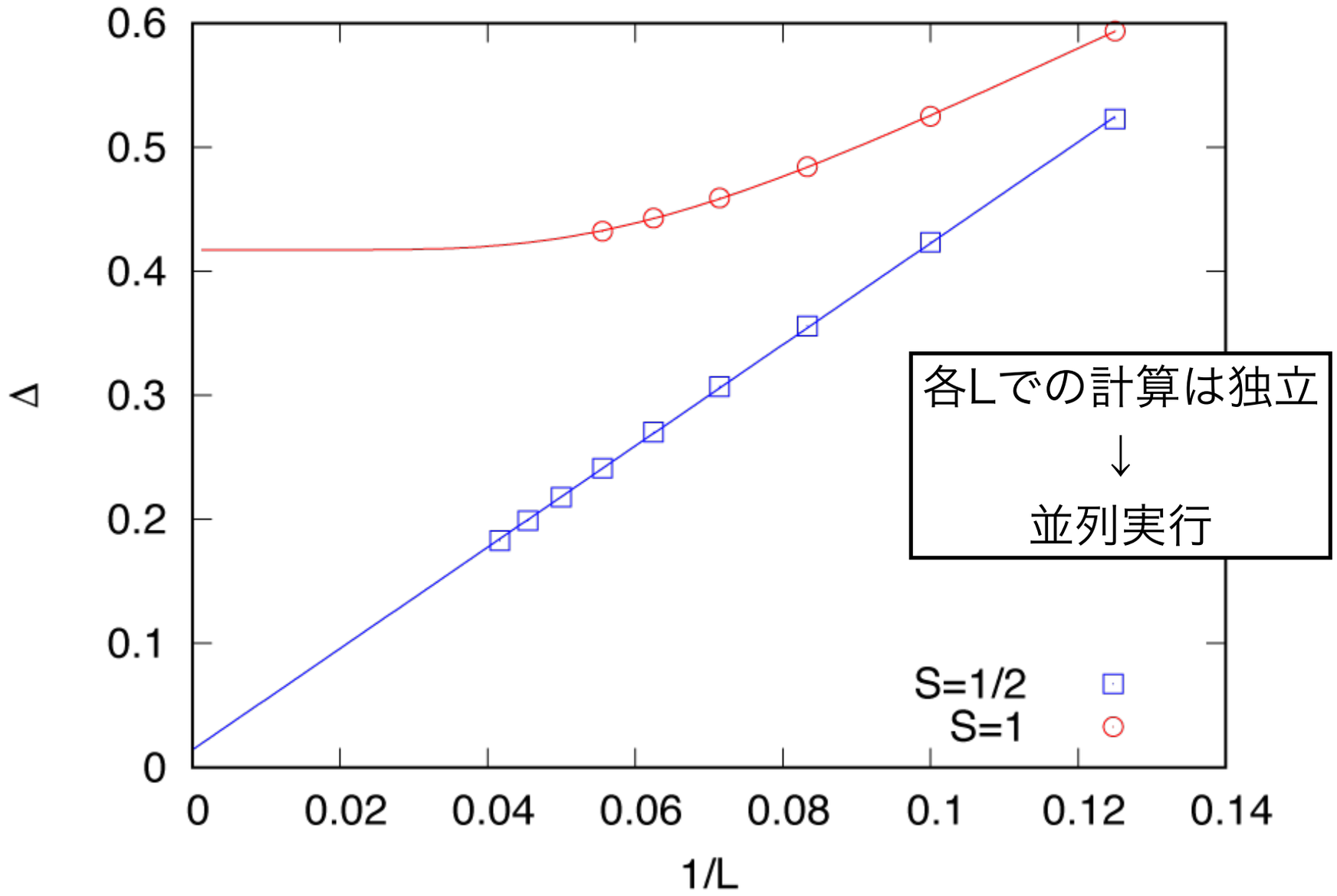
- **moller**とは、

> 「スパコン上でプログラムをバルク実行する
> ジョブスクリプト」を生成するツール

- スパコンのフロントエンド上、またはオフラインで実行する
- YAML形式(構造化テキスト)でジョブの内容を記述
- バルク実行に関するプラットフォーム依存な詳細を隠蔽
- マルチプラットフォーム (現在は物性研スパコン ohtaka, kugui に対応)

mollerを使う

- 例として、厳密対角化ソフトウェア HΦ を用い、S=1/2 反強磁性ハイゼンベルク鎖の励起ギャップΔのシステムサイズL依存性を評価する。



mollerを使う

1. mollerの入力ファイル input.yaml を用意する

input.yaml

```
name: HPhi
description: AFH chain

platform:
  system:  ohtaka
  queue:   i8cpu
  node:    8
  elapsed: 00:30:00

prologue:
  code: |
    source /home/issp/materiapps/
oneapi_compiler_classic-2023.0.0--
openmpi-4.1.5/hphi/hphivars.sh

    ulimit -s unlimited
    export KMP_STACKSIZE=512m
    export UCX_TLS='self,sm,ud'

jobs:
  hphi:
    description: run HPhi
    node: [2,8,32]
    run: |
      srun HPhi -s stan.in
```

mollerを使う

1. mollerの入力ファイル input.yaml を用意する

input.yaml

```
name: HPhi
description: AFH chain

platform:
  system:   ohtaka
  queue:    i8cpu
  node:     8
  elapsed:  00:30:00

prologue:
  code: |
    source /home/issp/materiapps/
oneapi_compiler_classic-2023.0.0--
openmpi-4.1.5/hphi/hphivars.sh

    ulimit -s unlimited
    export KMP_STACKSIZE=512m
    export UCX_TLS='self,sm,ud'

jobs:
  hphi:
    description: run HPhi
    node: [2,8,32]
    run: |
      srun HPhi -s stan.in
```

バッチジョブの設定

各タスクの並列度

single版の job.sh

```
#!/bin/sh
#SBATCH -p i8cpu
#SBATCH -N 2
#SBATCH -n 8
#SBATCH -c 32
#SBATCH -t 00:10:00

source /home/issp/materiapps/
oneapi_compiler_classic-2023.0.0--
openmpi-4.1.5/hphi/hphivars.sh

ulimit -s unlimited
export KMP_STACKSIZE=512m
export UCX_TLS='self,sm,ud'

srun HPhi -s stan.in
```

プラットフォーム固有の設定

計算内容

mollerを使う

2. moller の実行環境をセットアップする

実際は1行

```
$ source /home/issp/materiapps/
oneapi_compiler_classic-2023.0.0--openmpi-4.1.5/moller/
mollervars.sh
```

(物性研システムB(ohtaka)の場合)

3. moller を実行してジョブスクリプトを生成する

```
$ moller input.yaml -o job.sh
```

- ジョブスクリプト job.sh が生成される

4. リストファイルを作成する

list.dat

```
L_8
L_10
L_12
L_14
L_16
L_18
L_20
L_22
L_24
```

- パラメータセットごとに個別のディレクトリを用意する
- ディレクトリのリスト list.dat を作成する

mollerを使う

ジョブ投入コマンドは
システムによる

5. バッチジョブを投入する

```
$ sbatch job.sh list.dat
```

6. 各ジョブのステータスは

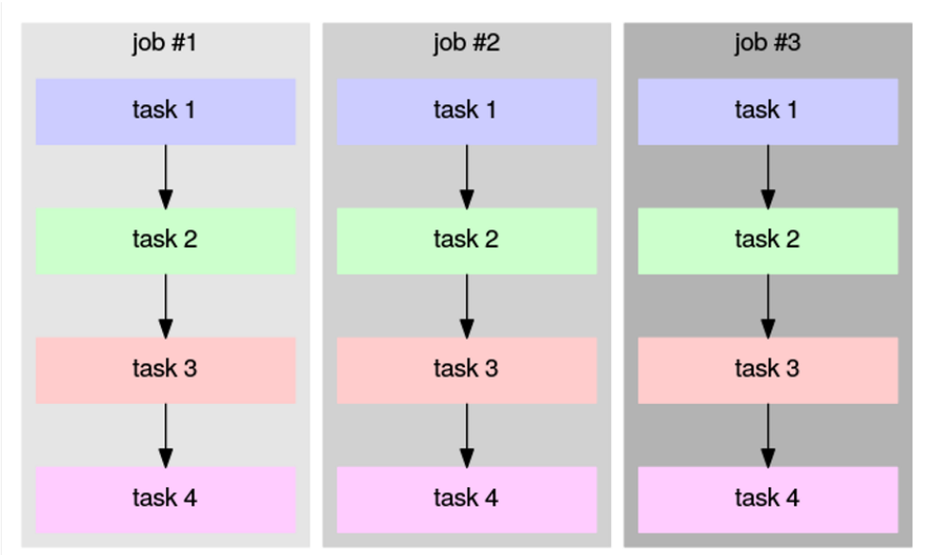
```
$ moller_status input.yaml list.dat
```

で確認できる。

```
| job    | hphi     |
|--------|----------|
| L_8    | o        |
| L_10   | o        |
| L_12   | o        |
| L_14   | o        |
| L_16   | o        |
| L_18   | o        |
| L_20   | o        |
| L_22   | o        |
| L_24   | o        |
```

mollerの機能

- 一つのパラメータセットに対する計算("ジョブ")を複数の"タスク"で構成できる
  - "タスク"ごとにジョブを並列して実行する
  - "逐次タスク"(ジョブによらない処理)を指定可
- 前のタスクが失敗した場合は、後続のタスクは実行されない
  (ジョブごと。他のジョブは影響を受けない)



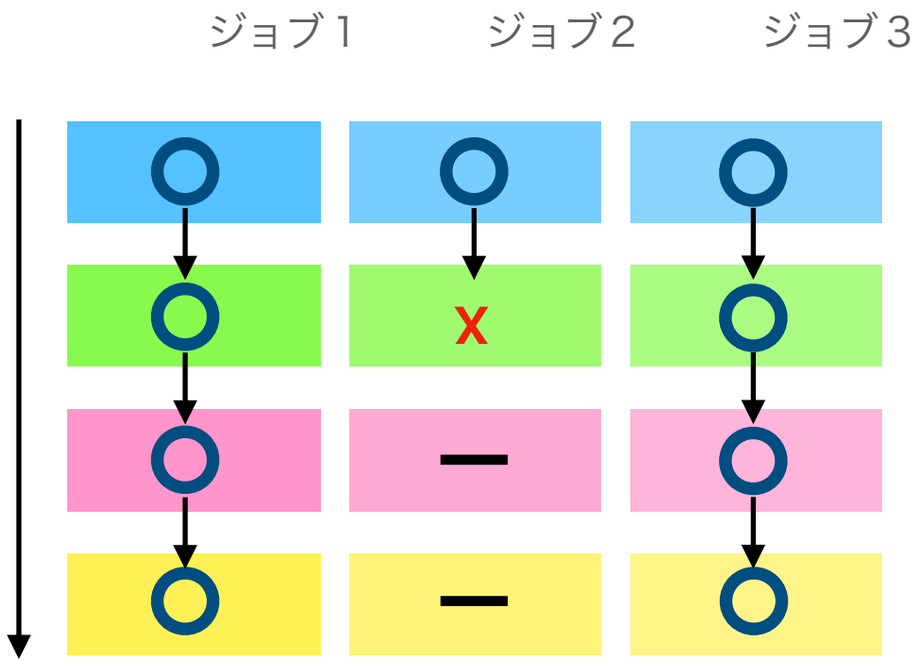
mollerの機能

- レジューム・リトライ
  - バッチジョブが途中で終了した場合、未処理のタスクから実行を継続する(レジューム)

```
$ sbatch job.sh list.dat
```

  - 失敗したタスクを再実行する(リトライ)

```
$ sbatch job.sh --retry list.dat
```



mollerをインストールする

- moller はオープンソース・ソフトウェアとして公開
  - GitHub リポジトリから取得・インストールできる

```
$ git clone https://github.com/issp-center-dev/moller.git
$ cd moller
$ python3 -m pip install .
```

- system = default を指定すると、一般のクラスタ計算機(ジョブスケジューラなし)で動作するスクリプトを生成

mollerをインストールする

- 現状のプラットフォーム対応状況
  - ジョブスケジューラ:
    - SLURM, PBS系 (PBS Pro, Torque), NQSV
    - ToDo: Fujitsu, (SGE)
  - MPIランタイム
    - OpenMPI, Intel MPI
    - ToDo: MPICH, MVAPICH, etc
  - (アクセラレータは別途対応が必要)
- 対応プラットフォームを拡充予定
  - 国内の主要なスパコンサイトへの展開

プリインストール済み

対応プラットフォーム

- 物性研究所
  - システムB ohtaka (SLURM)
  - システムC kugui (PBS Pro)
- クラスタ計算機 (Torque)

(対応中)

- 東北大 AOBA-S/A/B (NQSV)

mollerの情報まとめ

- moller ウェブサイト
  - 物性研ソフトウェア開発高度化ウェブサイト
    https://www.pasums.issp.u-tokyo.ac.jp/htp-tools/
- ソースコード・リポジトリ
  - GitHubリポジトリ
    https://github.com/issp-center-dev/moller
- オンライン・マニュアル
  - https://www.pasums.issp.u-tokyo.ac.jp/htp-tools/doc/manual/
- サンプル・チュートリアル
  - 物性研データリポジトリ: Moller Gallery
    https://datarepo.mdcl.issp.u-tokyo.ac.jp/repo/38
- 問い合わせ先
  - GitHub Issue　https://github.com/issp-center-dev/moller/issues
  - HTP-tools開発グループ　htp-tools-dev@issp.u-tokyo.ac.jp

HTP-tools (cif2x + moller)

機械学習を活用した物性予測や物質設計(マテリアルズ・インフォマティクス)

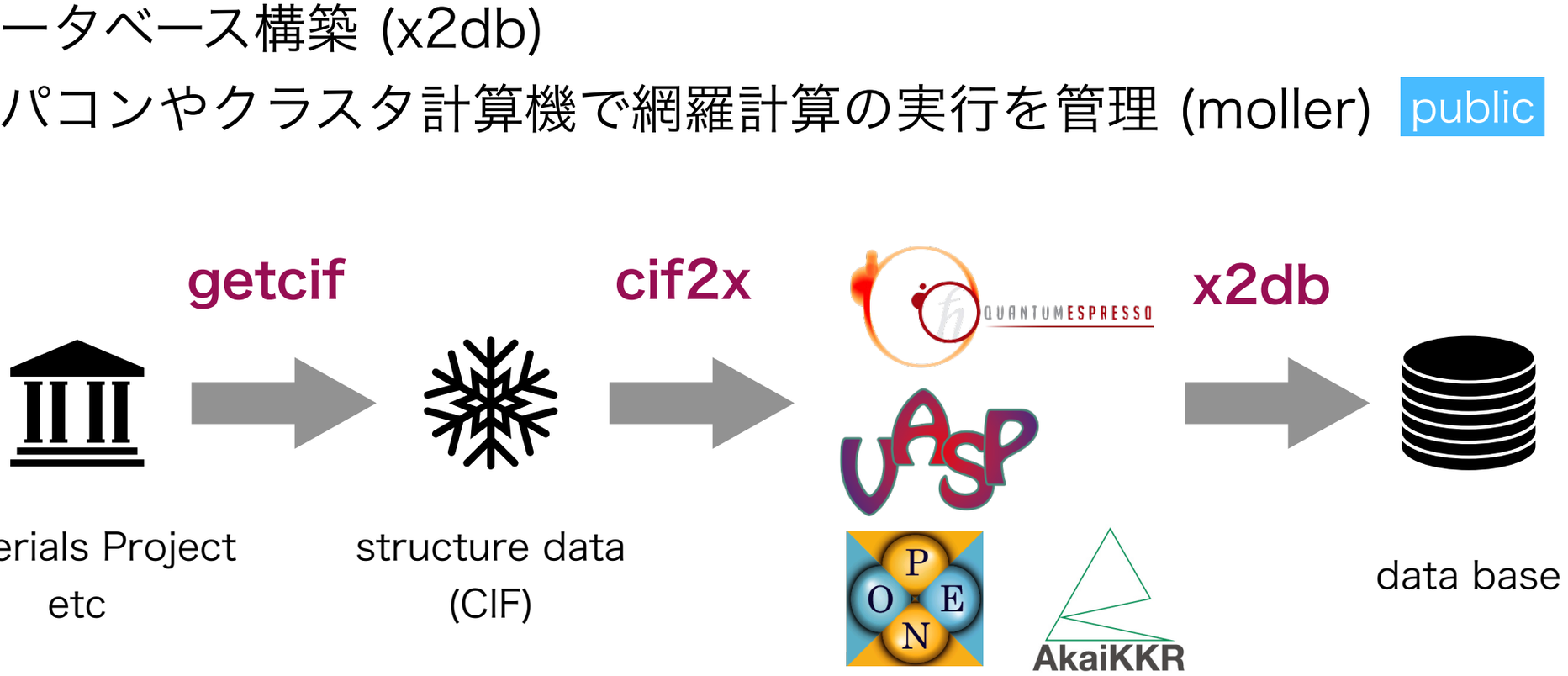
- 機械学習の精度は適切な教師データの準備に大きく依存
- 迅速に大量の教師データを生成するためのツールや環境を整備

大型計算機上での網羅的な第一原理計算の実行を支援するツールの開発

- High-ThroughPut (HTP) tools
  - 2023年度 物性研ソフトウェア開発高度化課題として実施
- 軽量のツールセット
  - 専用マシンや複雑なセットアップなしに、容易に目的のデータセットを構築できるようにする
- プロジェクトサイト
  - https://www.pasums.issp.u-tokyo.ac.jp/htp-tools/

HTP-toolsの構成

① 公開データベースから結晶構造データを検索・一括取得 (getcif) public
② 第一原理計算ソフトウェアの入力データを生成 (cif2x) public
③ データベース構築 (x2db)
④ スパコンやクラスタ計算機で網羅計算の実行を管理 (moller) public



bulk execution