京を利用したものづくり

小野　謙二
理化学研究所 計算科学研究機構
/ 東京大学 生産技術研究所

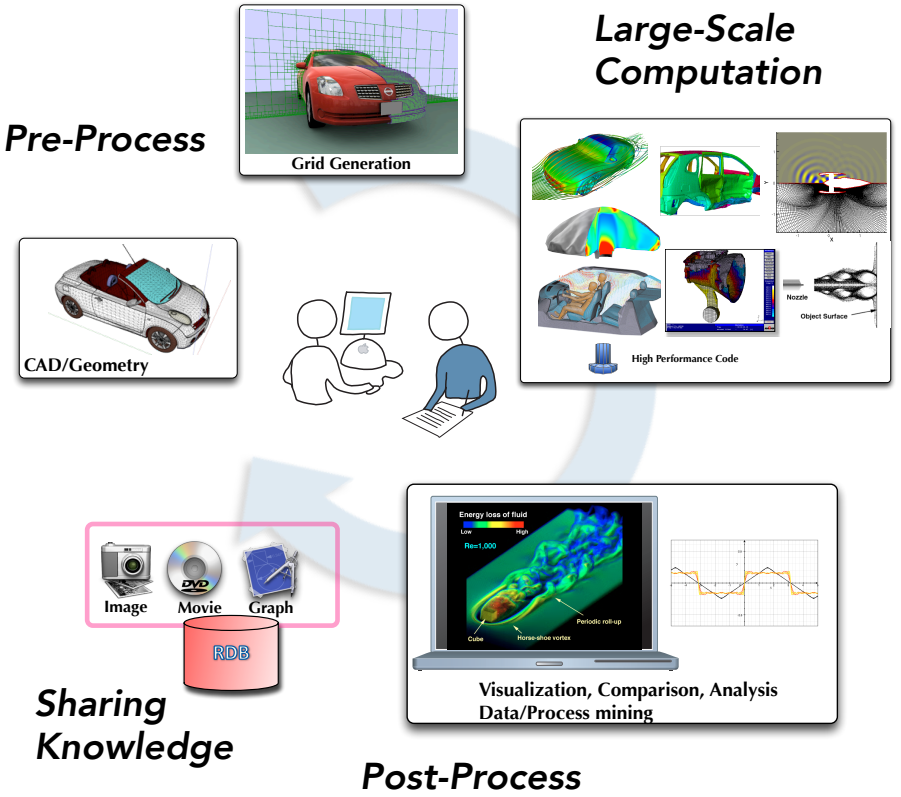
第5回HPCI戦略プログラム合同研究交流会

TOC

- 大規模格子生成技術
  - 格子生成＝交点計算＋スキーム開発
- ファイルハンドリング
  - 大規模ファイル管理ライブラリ
- プロダクトランの支援
  - 検証事例
  - 実行支援のしくみ
  - アウトリーチ

産業分野の大規模計算と活用の問題点

- 格子生成
  - 大規模な格子をどのように作るか？
- 並列計算
  - シミュレータの並列化
  - ファイル入出力
  - ユーティリティ
- 可視化とデータ処理
  - 多数のファイルから効率よくイメージを作る
  - 効率よくデータ処理を行う
- データハンドリング
  - 多数のファイルの管理
- 自動処理
  - 定型処理の自動化
- プロジェクト管理
  - シミュレーションケースの管理
- 利活用
  - 結果の有効利用

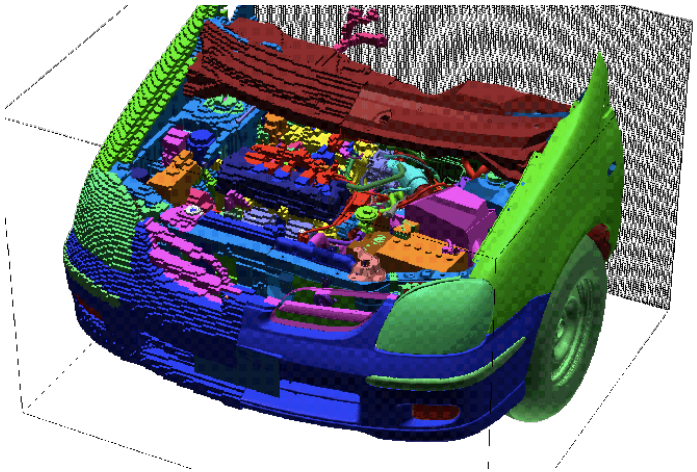


大規模計算における課題
チャレンジ

1. 大規模な計算モデルをどうやって作るか
2. 如何に短時間で結果を得るか
3. 計算結果のファイルをどのように効率的に扱うか
4. 計算結果をどのように可視化・データ処理するか
5. 結果を有効活用する方法は？

前処理から自動生成へ

従来

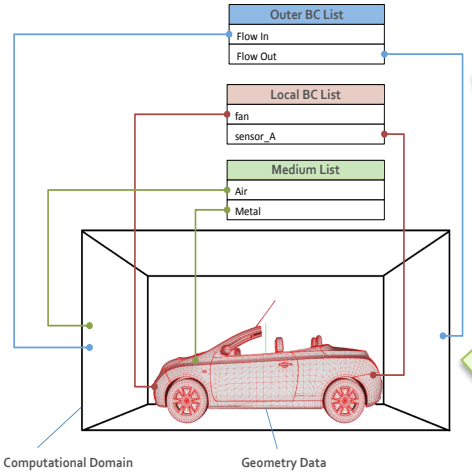


- 計算の前に，PCで格子を作成
- オペレータが作業（品質がばらつく）
- ファイル転送時間

並列領域分割に対応した格子ファイルを予め作成しておく（予め決められた分割数のみでしか計算できない）

並列計算

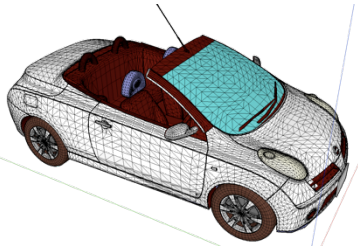
自動生成




```
DomainInfo {
  Global_origin   = (-0.5, -0.5, -0.5   )
  Global_region   = (1.0,  1.0,  1.0    )
  Global_voxel    = (64    , 64    , 64    )
  Global_division = (1     , 1     , 1     )
  ActiveSubDomain_File = "hoge"
}
```

計算領域と境界条件は前処理で指定（テキストファイル）

格子の並列自動生成に必要な最低限の情報を準備

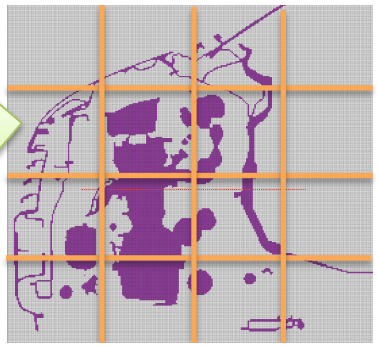


形状データ

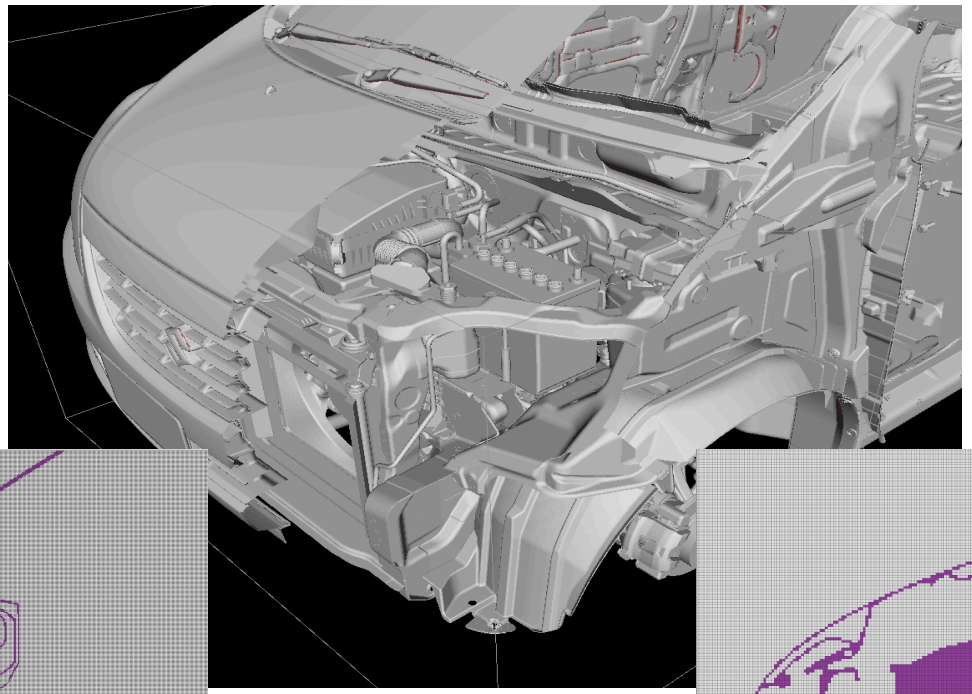
並列処理

ロバストな格子自動生成アルゴリズム

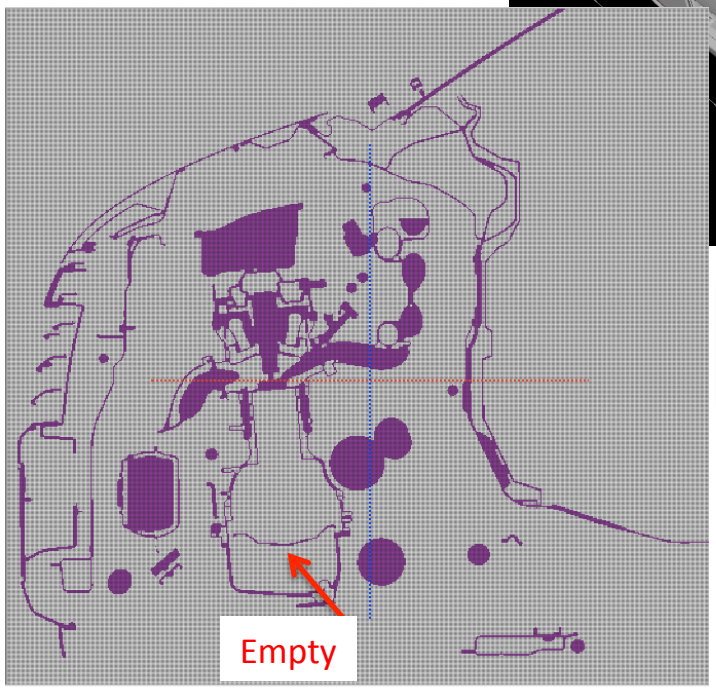
- 一定品質
- 短時間
- 直交（階層型）で可能に



Computation Model of 29G Grid from Practical Geometry Data



2mm



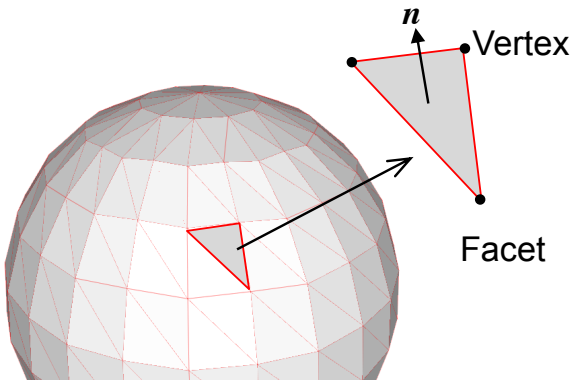
4mm

If a hole exists, fill algorithm does not work

- Water-tight STL
- Refine a fill algorithm

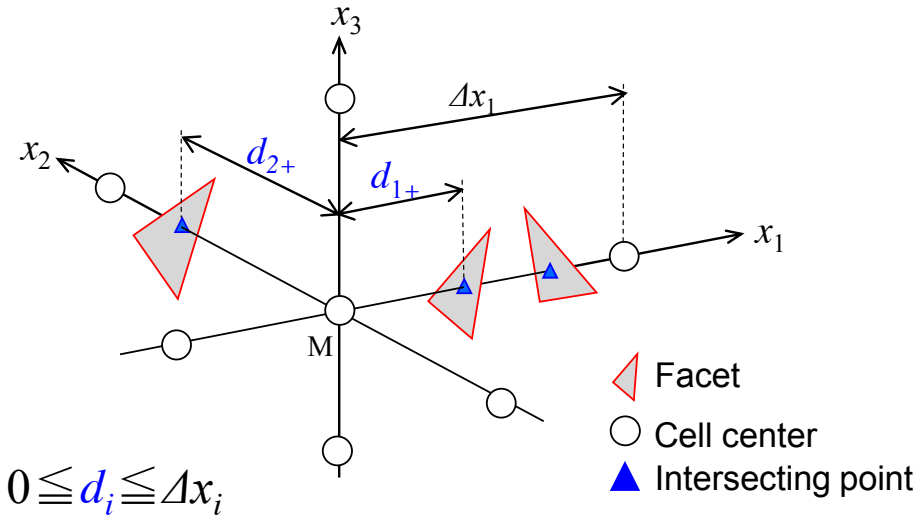
距離情報

STL(STereo-Lithography)形式を前提とする
facetとセルセンタ間を結ぶ線との交差判定を行い，距離情報d_iを取得



Geometry

An example of STL data.



$0 \leqq d_i \leqq \Delta x_i$

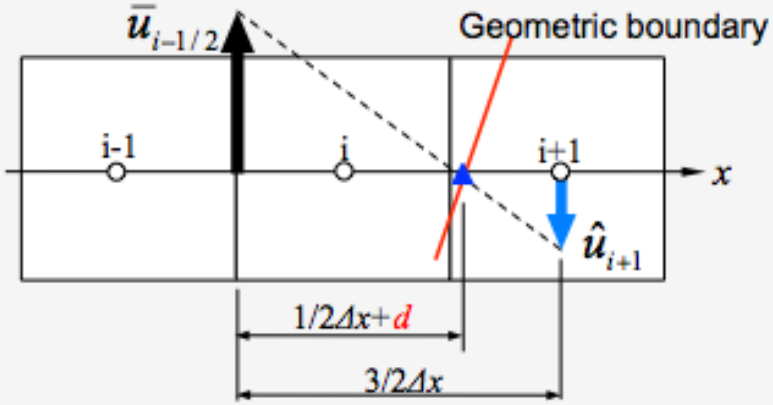
・交差判定はCG分野のレイトレーシングの技術を利用
→プログラムによる自動化が可能

・非水密なポリゴンデータ，薄い板(シェル)からでも距離情報を取得可能

Improvement of Shape Representation

Calculation under two assumptions

- Gradient of velocity is Liner in the vicinity of the geometric boundary
- Non-slip condition on the geometric boundary



$$\hat{u}_{i+1} = \left(1 - \frac{3\Delta x}{\Delta x + 2d}\right)\bar{u}_{i-1/2}$$

$\bar{u}_{i-1/2}$ linearly interpolated velocity using u_i and u_{i-1}

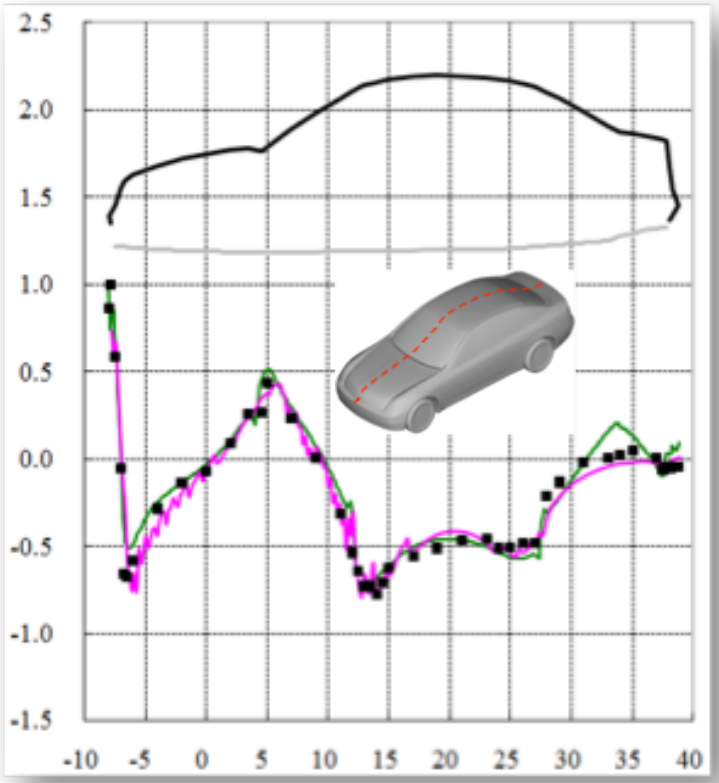
Discretization of the RHS of the Poisson equation

- using the extrapolated velocity $\hat{u}_{i+1}$ in the same manner

$$\frac{\partial}{\partial x_j}\left(\frac{\partial p^{n+1}}{\partial x_j}\right) = \frac{1}{\Delta t}\frac{\partial u_j^*}{\partial x_j}$$

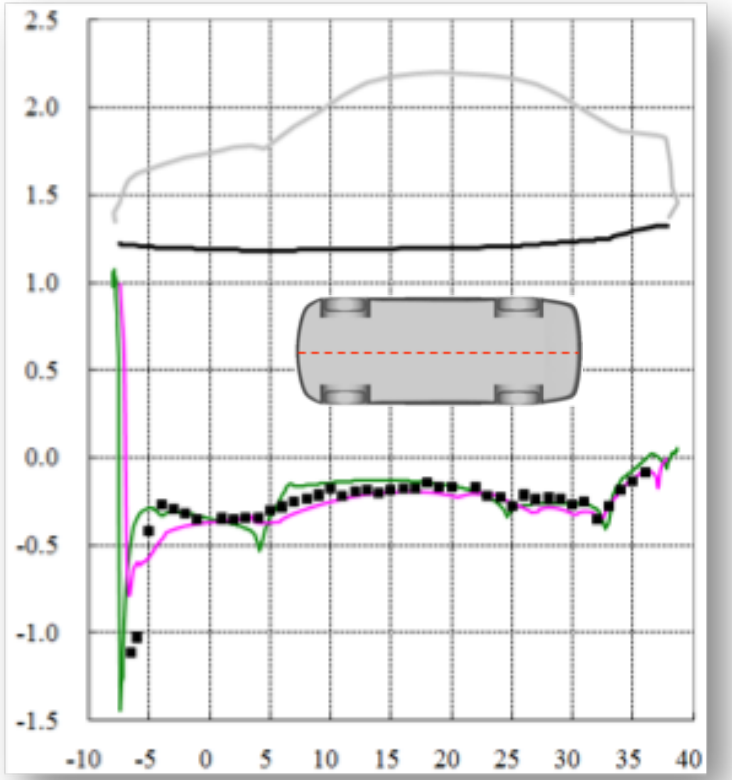
Comparison of pressure distribution

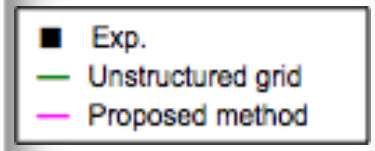
Upper-body side



Cp

Under-body side





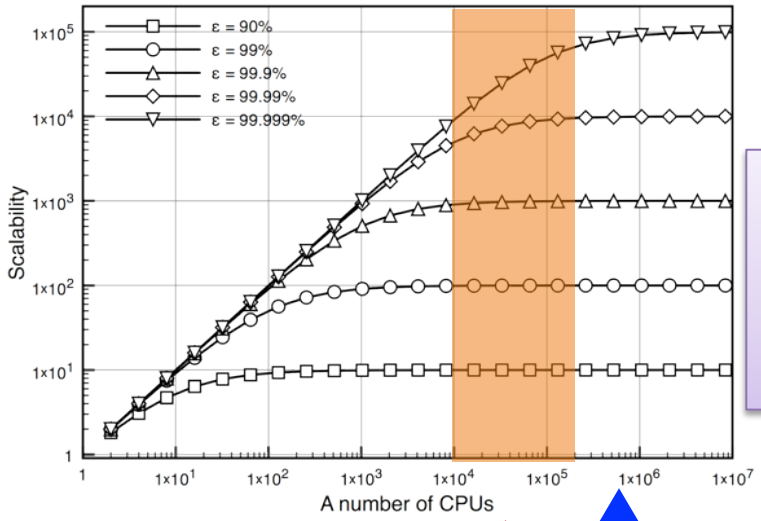
Computational condition

Car Length	L=0.903 [m]
Inlet velocity	u_{in}=25 [m/sec.]
Reynolds number	Re=1.5x10^6 (Based on U_{in}, L)

プログラム開発・チューニング

Amdahl's law

- 並列性能の指標

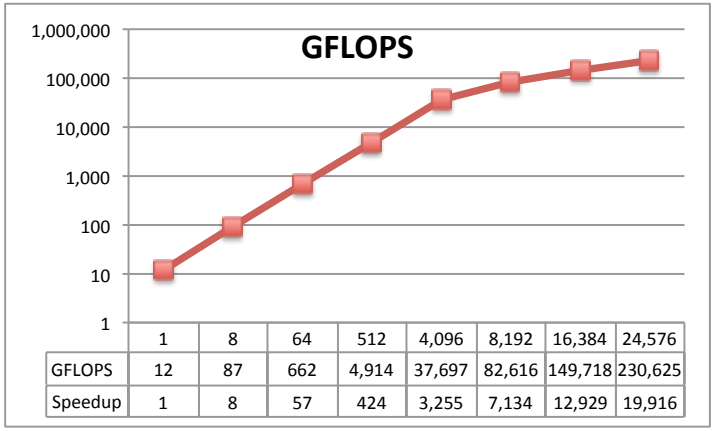


- Hybrid並列
- SIMD化
- データ構造

並列化率99.999%のとき，スピードアップの最大は1,000倍．2000CPUでも1000倍が上限．

FrontFlow / violet

- Weak Scaling
  - 200万セル/node
  - 9.38 Gflops/24576ノード （7.3%）



	1	8	64	512	4,096	8,192	16,384	24,576
GFLOPS	12	87	662	4,914	37,697	82,616	149,718	230,625
Speedup	1	8	57	424	3,255	7,134	12,929	19,916

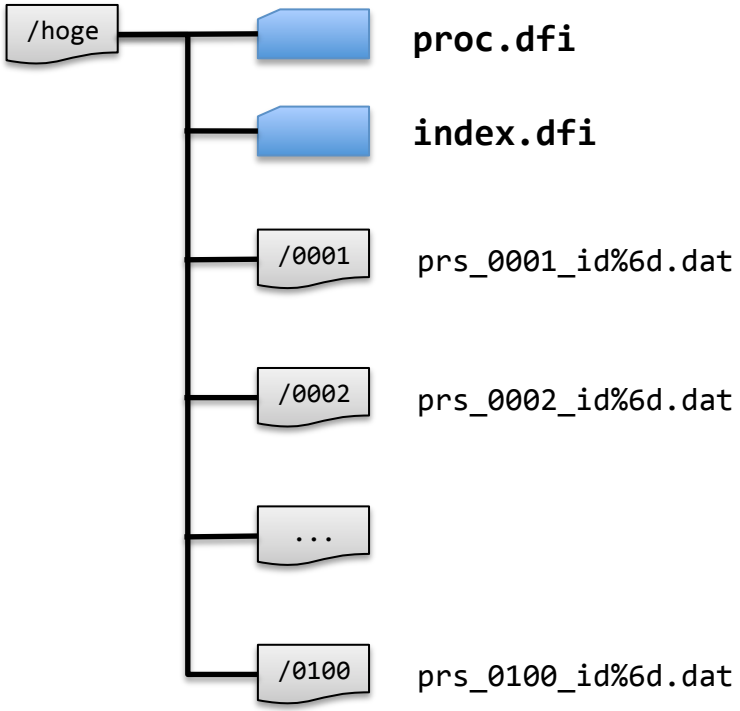
- Strong Scaling
  - 5万セル/node
  - 6.18 Gflops/9216ノード （4.8%）

大規模ファイル管理ライブラリの主要な機能

- 分散ファイル管理
  - 性能 > 各プロセスが個別に入出力 スケーラビリティを確保
  - 複数のファイルを管理するメタ情報
  - 複数ファイルをディレクトリで管理（各ステップ毎など）
  - エンディアンフリー
  - シミュレーションからデータ処理にわたるデータ管理
- M x Nデータロード
  - Mプロセスの出力ファイルをNプロセスにロードする
- リファインメントデータロード
  - 粗い格子で計算した結果を1段細かい格子にマッピングする
- ステージング対応機能
  - スクリプトで記述できない場合の汎用的な対応
- 将来的にはストリーミング対応フォーマットへの拡張
  - インタラクティブ可視化用途、LOD or プログレッシブなデータ処理対応

基本的なファイル管理方式

- 並列、時系列データを管理
  - 管理用メタファイル
    - proc.dfi
    - index.dfi
  - 出力ディレクトリ構造
- 以下の機能をサポート
  - MxNリスタート
  - リファインメントリスタート
  - ステージング



proc.dfi

```
// ドメイン情報
Domain {
 GlobalOrigin   = (-3.00, -3.00, -3.00) // 計算空間の起点座標
 GlobalRegion   = ( 6.00,  6.00,  6.00) // 計算空間の各軸方向の長さ
 GlobalVoxel    = (64, 64, 64)          // 計算領域全体のボクセル数
 GlobalDivision = (1, 1, 1)             // 計算領域の部分領域の分割数
}

// 並列情報
MPI {
  NumberOfRank  = 128  // プロセス数
  NumberOfGroup = 1    // グループ数
}

// 各プロセスの情報
Process {
  Rank[@] {
    ID         = 0                  // ファイル出力時のランク番号
    Hostname   = “Iridium.local”    // 計算ノードのホスト名
    VoxelSize = (64, 64, 64)        // 各領域のボクセルサイズ
    HeadIndex = (1, 1, 1)           // 各領域の開始インデクス
    TailIndex = (64, 64, 64)        // 各領域の終了インデクス
  }
  ...
}
```

ホスト名は必須では無い

index.dfi

```
// ファイル情報
FileInfo {
  DirectoryPath = “./”      // ファイルの存在するディレクトリ
                               (DFFファイルからの相対パス)
  Prefix        = "vel"     // ベースファイル名       ※1
  FileFormat    = "bov"     // ファイルタイプ、拡張子 ※1
  GuideCell     = 0
  DataType      = "Float32” // データタイプ           ※2
  Endian        = "little"  // データのエンディアン   ※3
  ArrayShape    = "ijkn"    // 配列形状               ※4
  Component     = 3         // 成分数(スカラーは1)    ※4
}

// ファイルパス情報
FilePath {
  Process      = “proc.dfi”
}
```

```
// 単位系 必要に応じて追加
Unit {
  Length   = ”M"   //(NonDimensional, m, cm, mm)
  L0       = 1.0   // 規格化に用いた長さスケール
  Velocity = ”m/s" //(NonDimensional, m/s)
  V0       = 3.4   // 代表速度 (m/s)
  Pressure = “Pa”  // (NonDimensional, Pa)
  P0       = 0.0   // 基準圧力(Pa)
  DiffPrs  = 510.0 // 圧力差(Pa)
  Temperatur = “C” // (NonDimensional, C, K)
  BaseTemp = 10.0  // 指定単位
  DiffTemp = 35.0  // 指定単位
}
```

※1 ファイル名
[Prefix]_[ステップ番号:10桁]_id[RankID:6桁].[Extension]

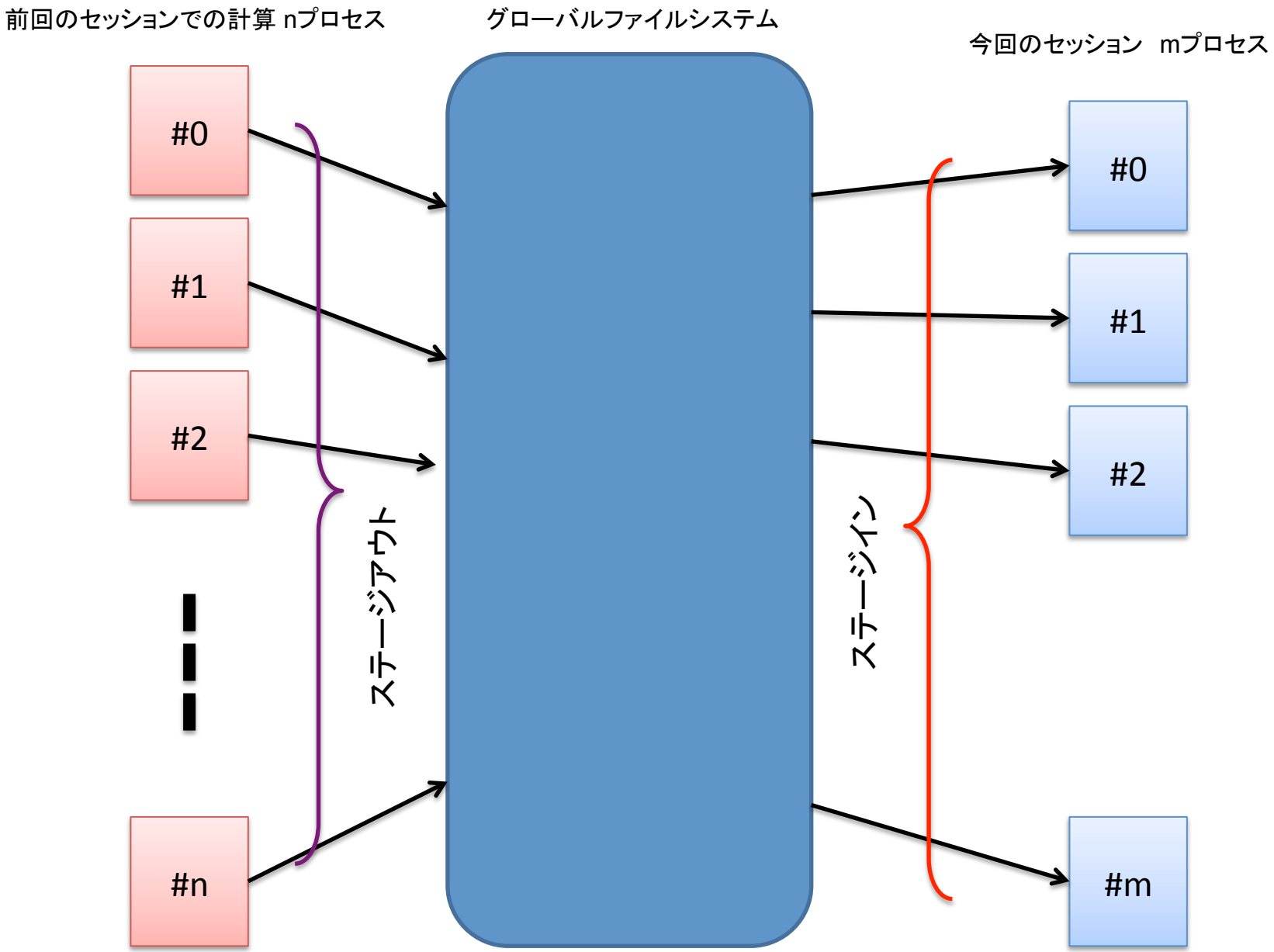
※2 Int8, UInt8, Int16, UInt16, Int32, Uint32, Int64, UInt64, Float32, Float64

※3 little, big, 省略時:実行プラットフォームと同じ

※4 ijkn, nijk
ijkn:(imax,jmax,kmax,Component)
nijk:(Component,imax,jmax,kmax)

index.dfi (contd.)

```
// 時系列情報
TimeSlice {
  Slice[@] { // ファイル出力回数分
    Step = 0
    Time = 0.0
    MinMax[@] {          // Component個
      Min  = -1.56e-2
      Max  = 8.2e-01
    }
    ... 任意のアノテーションが追加可能
 }
  ...
  Slice[@] {
    Step = 1000
    Time = 10.0
    MinMax[@] {          // Component個
      Min  = -8.5e-1
      Max  = 9.1e+01
    }
    ... 任意のアノテーションが追加可能
 }
}
```


前回のセッションでの計算 nプロセス
グローバルファイルシステム
今回のセッション　mプロセス
#0
#1
#2
#n
ステージアウト
ステージイン
#0
#1
#2
#m

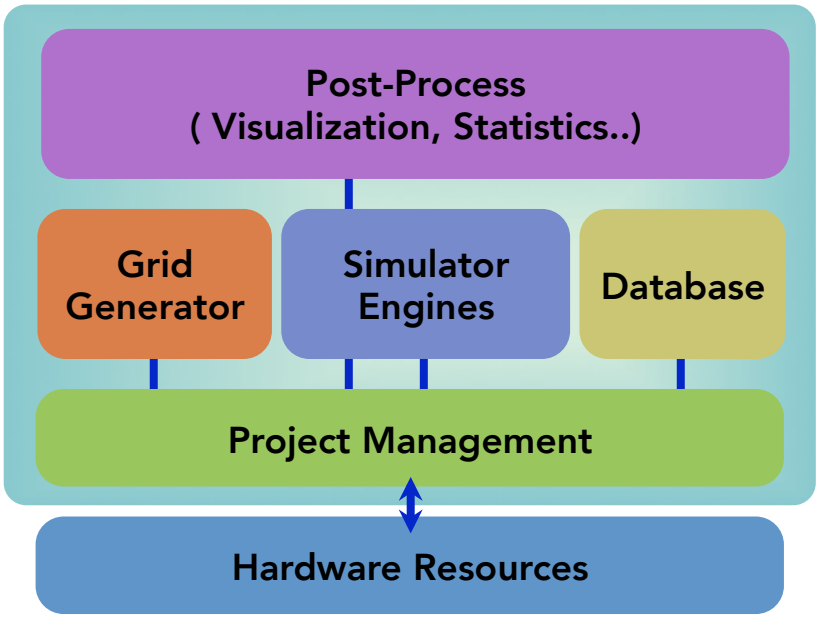
HPC/PFとは？

- ❖ 最先端の大規模並列シミュレーション技術を**研究/設計現場の道具**にする基盤ソフトウェア
- ❖ ソフトウェア利用環境の提供だけでなく、有用な実例を豊富に提供
- ❖ 企業内の小規模PCクラスタから世界最速スパコン「京」まで**多様な環境で運用可能**

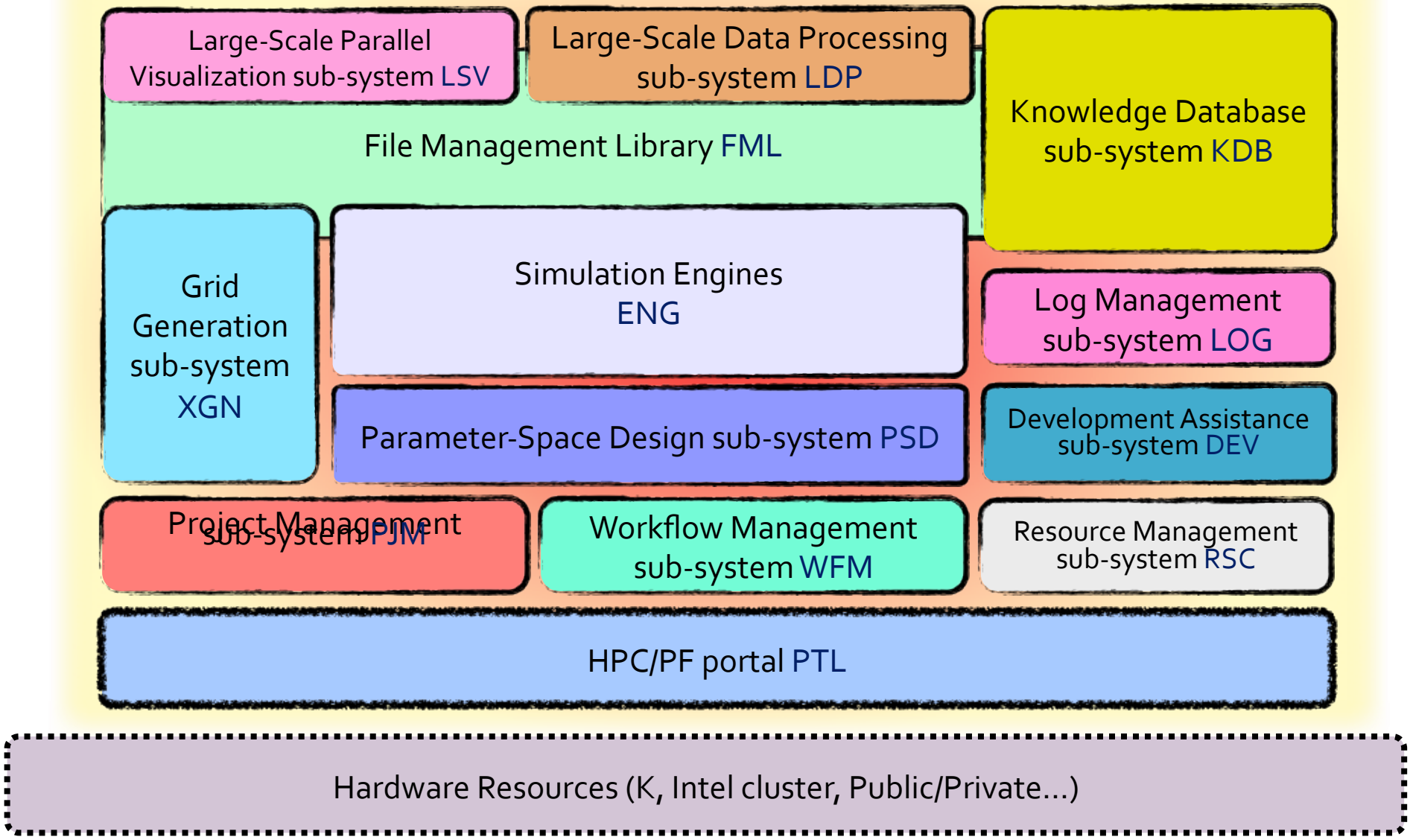
- High-Performance Computing Platform

- 大規模シミュレーションの効率的な実行を補助する実行ツールのパッケージ
  - 多くのツール（アプリ）群で構築
  - ツール間は緩い連携
    - アプリ間のAPIの設計が必要
    - 開発は各アプリ独立で可能
    - 各機能は入れ替え可能

- 設計活動の支援
  - 設計に必要な情報を作り出す



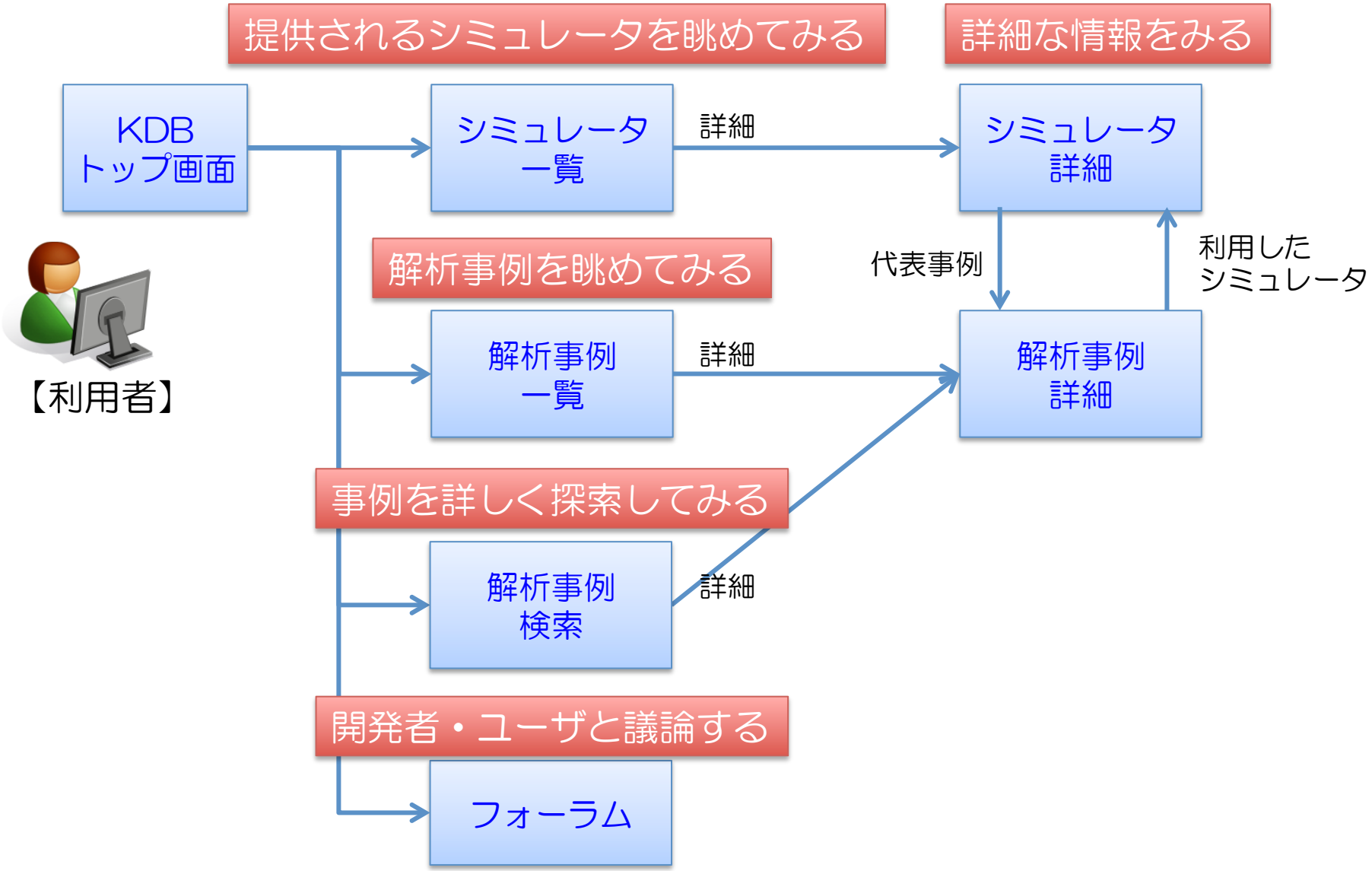
Structure of HPC/PF



事例検証DB

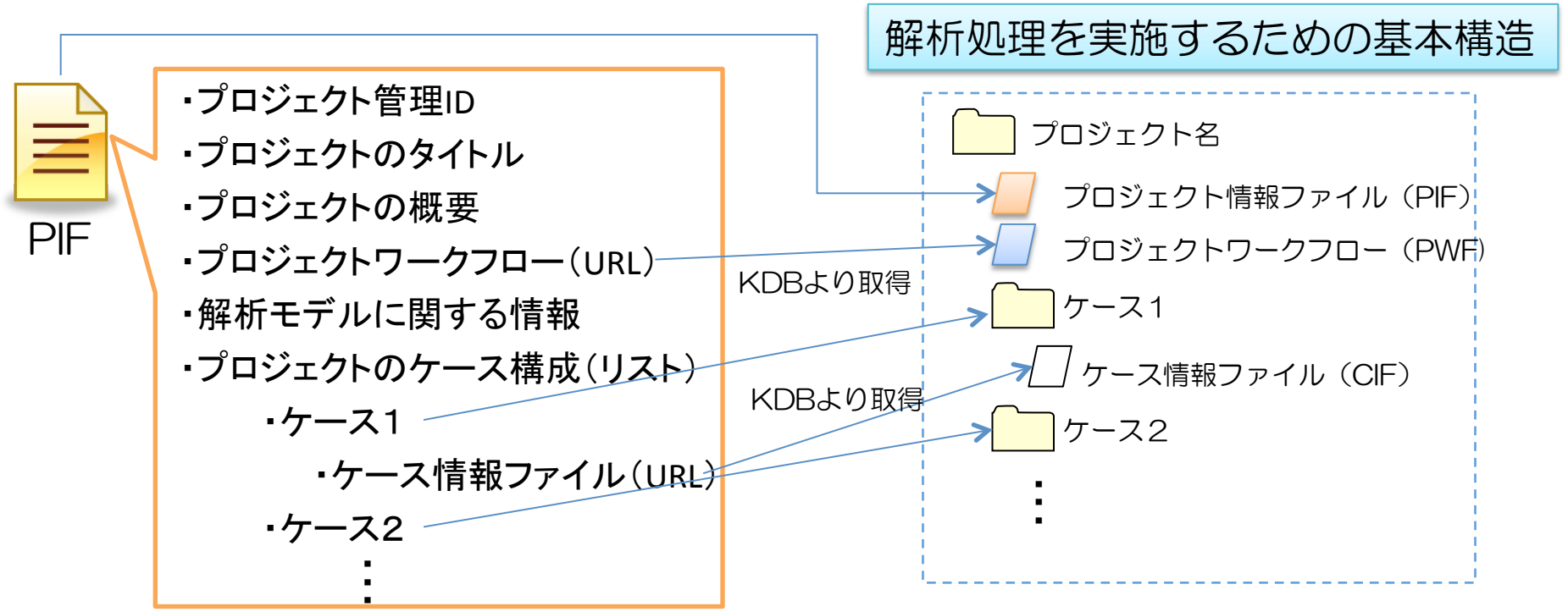
- シミュレータの情報
  - 特徴、機能説明
  - 解析可能なモデル規模
  - 代表的な解析事例
  - etc…
- シミュレータによる解析事例
  - V&V区分
  - 解析種別、物理現象、解析モデル
  - 実行環境、解析所要時間
  - 解析結果の詳細解説
  - 追体験可能なファイルセット
  - etc…

KDBの情報提供機能の構成

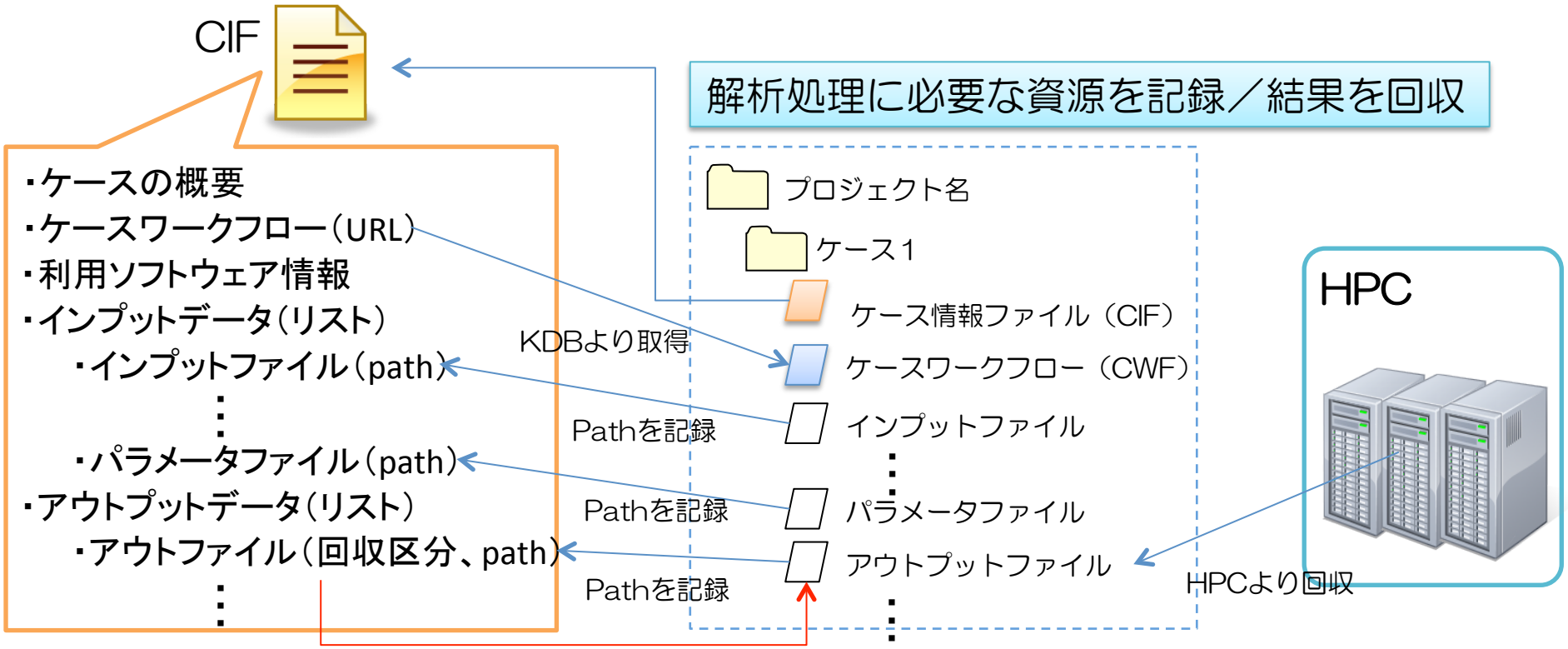


プロジェクト管理（PJM）

- プロジェクト内のリソース管理
  - 解析処理の実行に必要となるすべてのファイル、および計算結果や二次加工データなどを収集管理
- プロジェクトの準備
  - プロジェクト情報ファイル（PIF）に基づき、解析処理の準備を行う

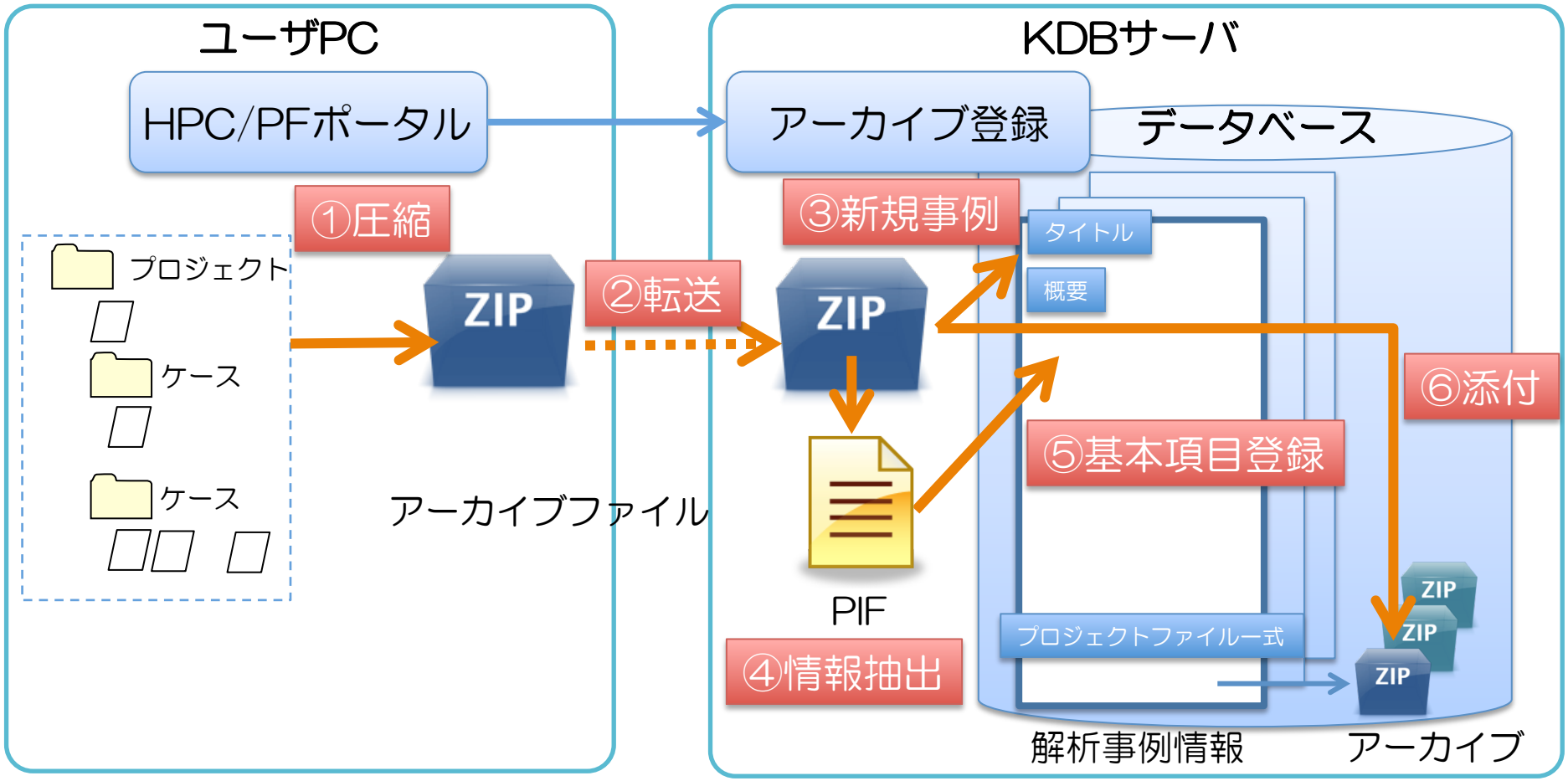


- ケースファイル
  - ケース情報ファイル（CIF）に基づき、解析（ケース）に必要となるファイルを準備し、保存場所（path）を記録する
  - HPC上で実行された解析処理結果ファイルについて、ケース情報ファイルに定義された回収区分に従いローカルPCへ回収し、保存場所（path）を記録する



データベースへの自動登録

- アーカイブによる基本項目登録
  - 解析事例情報を生成し、PIFより、基本項目を自動登録、アーカイブファイルを添付登録する



パラメータサーベイの枠組み

Xcrypt：並列ジョブ管理のためのPerlベースのスクリプト言語

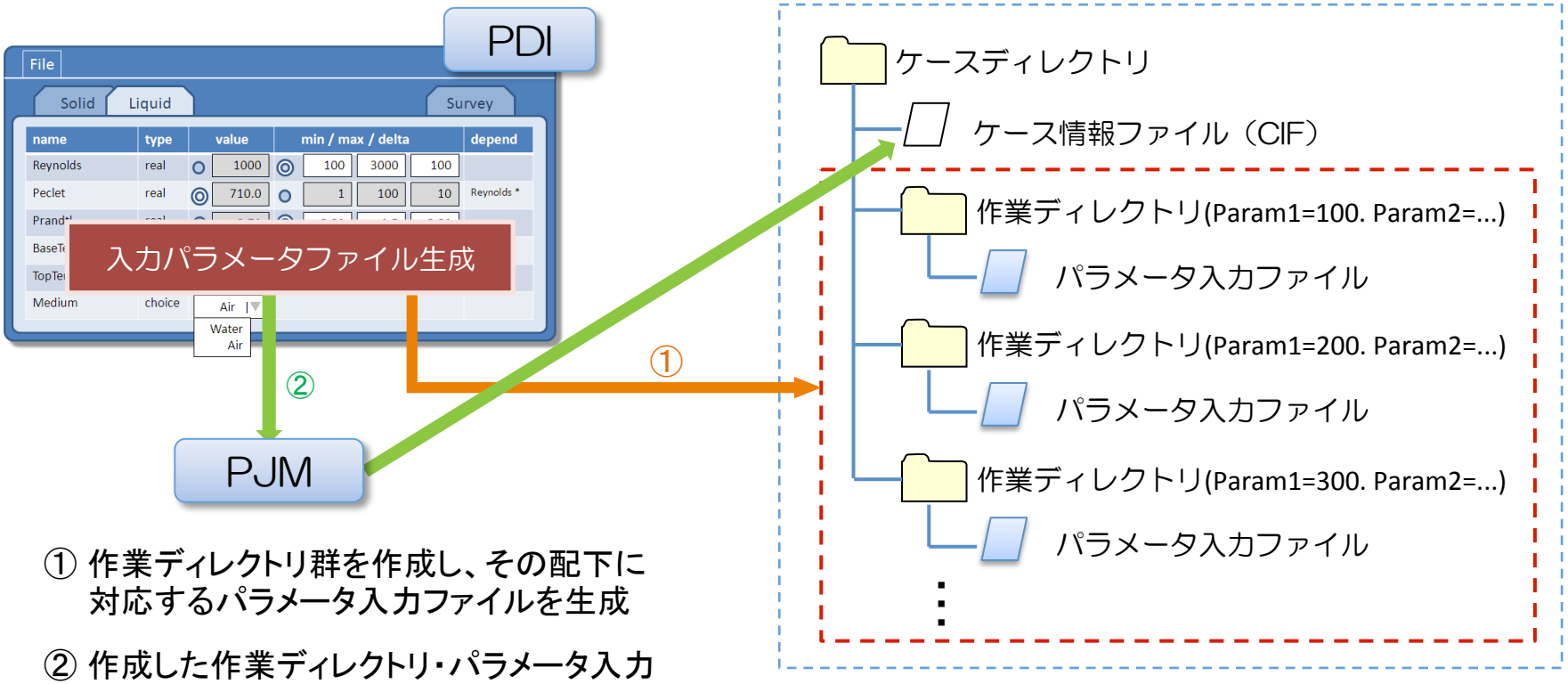
- 大量のジョブの投入と管理、待ち合わせ
- ジョブスケージューラ(NQS, Torque, SGE)の違いを吸収
  ジョブ投入を非同期手続き呼び出しとして扱える
- リモートファイル転送・リモートジョブ投入機能
- 各ケース毎のワークディレクトリ作成機能（Sandbox）
- Perlベースの手続き記述によるワークフローの構成
  ループ（while, foreach)、分岐(if, unless)等の記述

これらの機能を利用し、パラメータサーベイの枠組みを構築

- 全空間探索（パラメータスイープ）
- 全空間探索＋パラメータ空間の枝刈り
- パラメータ空間の絞り込み（遺伝的アルゴリズム等）

パラメータ空間設計

- PJMとの連携
  - 生成されたパラメータ群に対応するディレクトリ管理

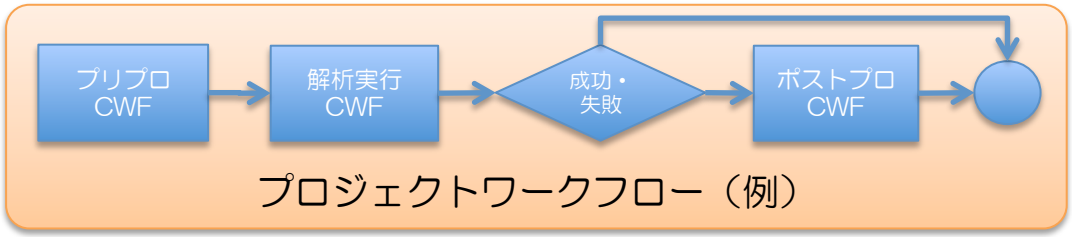


① 作業ディレクトリ群を作成し、その配下に
対応するパラメータ入力ファイルを生成

② 作成した作業ディレクトリ・パラメータ入力
ファイル群をPJMに登録依頼
（PJMは情報をケース情報ファイルに反映）

ワークフロー

- ワークフローの記述
  - 2つの粒度のワークフロー記述ファイルで構成
  1. プロジェクトワークフローファイル(PWF)
     - 解析プロジェクト全体の処理の流れを定義
     - シェルスクリプトで記述
  2. ケースワークフローファイル(CWF)
     - プリプロセス、解析実行、ポストプロセス、等の実行単位毎／アプリケーション毎の処理流れを定義
     - 基本はシェルスクリプトで記述。スパコンへのバッチジョブ投入はXcryptを利用
     - Xcrypt : 京大開発のジョブ並列実行支援記述言語（Perlベース）



アウトリーチに向けたシステム設計

- Webサイトの設計
  - 構成要素として、HPC/PFのKDB
  - エディトリアル　一般向けの記事

- 機能
  - 情報提供
  - ソース、パッケージ（外部サイト利用リンク）

アウトリーチサイトのプロトタイプ

まとめ

- 大規模計算のための要素技術開発
- 大規模計算トライアル
- プロダクトランを支援するシステム設計
- 情報発信のしくみを設計