シームレス高生産・高性能プログラミング環境

高生産並列スクリプト言語に関する研究

京都大学
学術情報メディアセンター
中島　浩

目次

- 研究開発の背景
  - スパコン計算の概念整理
- 研究開発の目的
  - 階層的大規模並列処理
  - 高生産＆並列＆スクリプト言語の意義
  - PDCA cycle 生産性向上
- 研究計画＆進捗
  - 解決すべき課題
  - 全体計画・全体構成
  - H20進捗＆H21/22計画

研究開発の背景
スパコン計算とパソコン計算

- 最近のスパコン≒パソコンの巨大な集合体
  - スパコン基本要素(プロセッサ＆メモリ)≒パソコン
  - スパコン基盤ソフトウェア(OS, コンパイラ)
    ≒パソコン基盤ソフトウェア
  - スパコン内部ネットワーク≒パソコン間ネットワーク
- スパコン性能 ≫ パソコン性能(x1000~100000)
  - スパコン規模(プロセッサ数, メモリ容量)
    ≫ パソコン(x数千~数十万)

  ➔パソコンの1万倍の性能のスパコンは …
    - 1万分の1の時間で仕事を片付ける (capability)
      and/or
    - 1万倍の仕事を同じ時間で片付ける (capacity)

研究開発の背景
Capability と Capacity

- Capability Computing
  - 1つの仕事を多人数で分担して片付ける
  - スパコン計算(並列計算)の王道
  - そもそも分担できるか／どう分担するかなど難度が高い
  - 10万スケールでは効果を得るのがさらに困難に

  ➔難度低減のための研究開発@筑波大・東大
- Capacity Computing
  - 多数(で同種)の仕事を人海戦術で片付ける
  - スパコン計算(並列計算)の覇道
  - 分担できること／分担の方法は自明 ➔ お手軽並列計算
  - 10万スケールでは手軽ではなくなってしまう

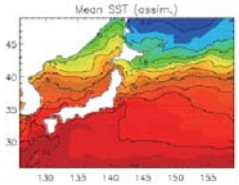
  ➔手軽さ維持のための研究開発@京大

研究開発の背景

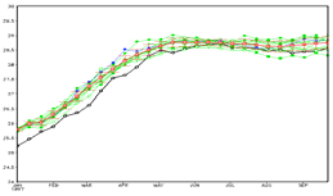
スパコン計算の典型例

例：気象・海洋シミュレーション

- 複雑・大規模な系を解析的にシミュレーション
  - 相互に絡み合った大量の計算を並列処理で高速化
    ➔ capability computing



- 大規模シミュレーションを大量に実行
  - 独立な計算を様々なパラメータ（初期条件など）で実行
    ＝アンサンブルシミュレーション
    ➔ capacity computing

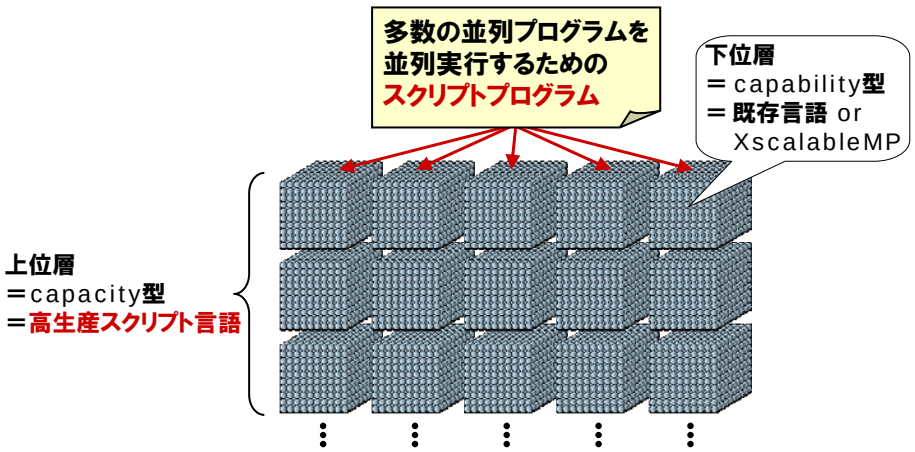


➔スパコン計算の**両輪**

研究開発の目的

階層的大規模並列処理

- Capability/Capacity の2階層化
  ➔1000並列 x 1000並列 ＝ 100万並列



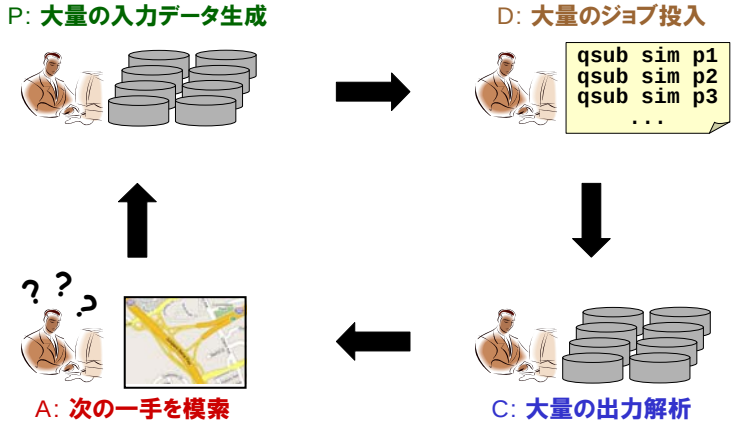
研究開発の目的
高生産＆並列＆スクリプト言語

- スクリプト言語
  - 元々は他のプログラム(群)の実行手順記述用の言語
    e.g. 1. シミュレーションプログラムを100個実行
    2. 結果の平均を算出するプログラムを実行
  - プログラム間をつなぐための便利な機能が豊富
    e.g. シミュレーション入力の雛形に必要なパラメータを埋め込み
    シミュレーション結果ファイルから所望の行を抽出
- 並列スクリプト言語
  - 他のプログラムの並列実行手順を記述する機能
    e.g. 1. シミュレーションプログラムを100個並列に実行
    2. 全てが終わったら結果の平均を算出するプログラムを実行
- 高生産並列スクリプト言語
  - プログラム間をつなぎを簡便に記述
    e.g. シミュレーション入力雛形へ簡単にパラメータを埋め込み
    シミュレーション結果から簡単に所望の行を抽出

研究開発の目的
最終目標＝PDCA cycle 生産性向上

- 高度なお手軽並列処理：e.g. パラメータ探索
  - 最適な答えを見つけるために色々なパラメータを試行

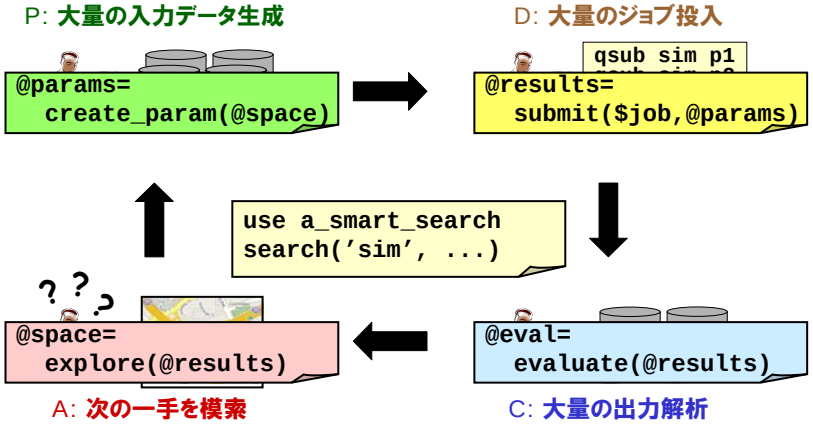
➔大量の実行＋知的な探索処理



研究開発の目的
最終目標＝PDCA cycle 生産性向上

- **高度なお手軽並列処理：** e.g. **パラメータ探索**
  - 最適な答えを見つけるために色々なパラメータを試行

  ➔**大量の実行＋知的な探索処理**



研究計画＆進捗
解決すべき課題 (1/2)

- **外部プログラム（バッチジョブ）制御の抽象化**
  - 普通の実行：直ちに実行開始＆終われば次を実行
  - **バッチジョブ実行**
    - スケジューラに（複数）プログラム実行（＝ジョブ）を（まとめて）依頼
    - スケジューラは（他のジョブも勘案し）適当なタイミング・順序で実行
    - マシンによってスケジューラは多種多様

  ➔実行機構の複雑さ＆多様さをユーザから隠蔽
- **外部プログラムインタフェースの抽象化**
  - ジョブの入力・出力はアプリケーションごとに多種多様
  - 入力生成・出力解析には共通点が多い

    e.g. 入力パラメータは正規分布からn個抽出

    出力に対する統計処理（平均，分散，検定など）

  ➔入出力をオブジェクトに抽象化して共通処理を適用

研究計画＆進捗

解決すべき課題 (2/2)

- 探索アルゴリズムのライブラリ化
  - ジョブの入力・出力＝探索の出力・入力は多種多様
  - 高度な探索アルゴリズム設計・実装はユーザには困難

  ➔探索の入出力をオブジェクトに抽象化

  ➔多様な探索法をプログラム部品(ライブラリ)として用意
- スクリプト言語レベルのデバッグ機能
  - プログラムは簡単でもジョブの実行時間・必要資源は膨大

    ➔間違ったスクリプト実行による損失は膨大
  - 簡単なプログラムでも間違いは生じうる

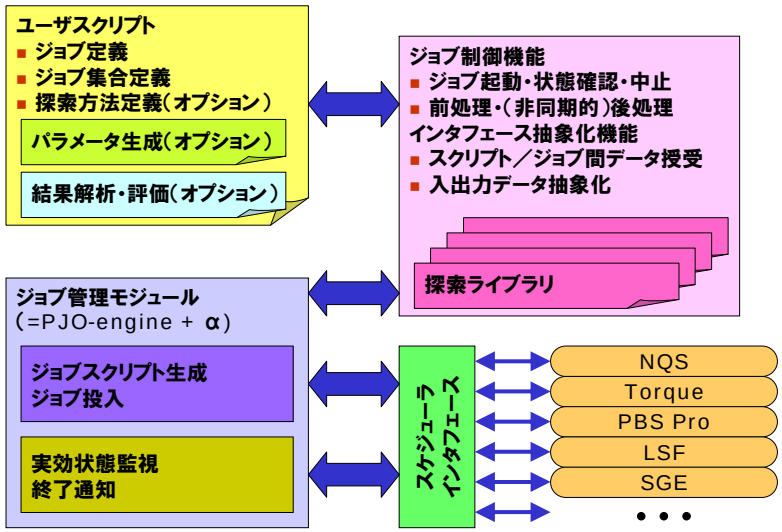
  ➔ジョブ実行を省略・簡略化してスクリプトだけをデバッグ

研究計画＆進捗

全体計画

- H20: 言語基本仕様設計＆試験実装
  - 研究開発全体の土台となる基本仕様を設計
  - 主要な機能を試験的に実装して仕様の妥当性を確認
- H21: プロトタイプ開発
  - 必要な機能を一通り実装
  - 全体として使える形に仕上げる
- H22: プロトタイプ評価＆拡張・改良設計＆実装
  - 多数の実例を用いて機能・性能を評価 (e.g. 真に簡単?)
  - 評価結果に基づき機能の追加・改良 (e.g. より簡単に)
  - プロトタイプには盛り込めない機能の追加 (e.g. デバッグ)
- H23: 最終版実装＆総合評価

研究計画＆進捗
全体構成



研究計画＆進捗
H20 進捗

- 言語基本仕様設計
  - 汎用＆標準的スクリプト言語 perl をベース
  - perlを拡張した多数ジョブ実行制御システムPJOを活用
- ジョブ制御抽象化
  - PJOジョブ制御機構可搬化(外注)
  - 中間レベルの制御記述仕様＆制御モジュール設計(内作)
  - スケジューラ仕様調査＆インタフェース設計(内作→外注)
- ジョブインタフェース抽象化
  - オブジェクト指向に基づくシステムの枠組設計(内作)
  - PJOジョブ制御機構のインタフェース全面改訂(外注)
- 探索ライブラリ
  - 単純なパラメータスイープ(総なめ)の実装(外注)
  - 実応用に基づくPDCAの実例記述(内作→内作)

研究計画＆進捗
H21 計画

- ジョブ制御抽象化
  - エンドユーザ向け簡易インタフェース実装（外注）
  - スクリプト実行のバッチジョブ化設計（外注）
  - 探索用ジョブ制御機能（中断・再開など）の実装（内作）
- ジョブインタフェース抽象化
  - 入力生成・出力解析機能実装（内作）
  - エンドユーザ向け簡易インタフェース実装（外注）
- 探索ライブラリ
  - パラメータスイープ／アンサンブル／初期値探索ライブラリ実装（内作）
  - 実例（海洋シミュレーション，プラズマシミュレーション）検証（内作）

研究計画＆進捗
H22 計画

- プロトタイプ評価・改良（主に内作）
  - 実例ベースの機能検証
  - 実例ベースのユーザインタフェース改善
  - 実例ベースの入力生成・出力解析・探索モジュール充実
- 機能追加（主に外注）
  - スクリプトのバッチジョブ実行機能
  - スクリプトレベルデバッグ用のジョブ実行省略・簡略化機能