10. クラスタシステム構築に関する知識Ⅱ

1. 科目の概要

HPC (High Performance Computing)クラスタにおけるプログラミングに利用できる、HPF、OpenMP、MPI といった技術について解説する。さらに PC クラスタの構築方法や PC クラスタを管理するための SCore、その他のユーテリティも紹介する。

2. 習得ポイント

本科目の学習により習得することが期待されるポイントは以下の通り。

習得ポイント	説 明	シラバスの対応コマ
II-10-1. HPF (High Performance Fortran)の基礎	並列プログラミングを可能にするHPF (High Performance Fortran)を説明する。データを並列化するデータマッピング、処理を並列化する計算マッピングの概念を示し、簡単なHPFプログラミングについて解説する。	8
II-10-2. OpenMPによる並列プログラミング	並列プログラミング向けのライブラリであるOpenMPの概要とその利用方法を解説する。並列実行領域の確保、ワークシェアリング、データ環境、同期構文、実行時間数など各種のAPIを紹介し、簡単なサンプルによる具体的な利用法を示す。	8
II-10-3. MPIの基礎	HPCクラスタにおける並列プログラミングで一般的に利用される技術であるMPI (Message Passing Interface)の概要とプログラミングモデル、基本機能の概要と、MPIの先端的な特徴について解説する。	9,10,11
II-10-4. MPIによる並列プログラミング	MPIを用いた具体的な並列プログラミングの例を示す。並列プログラミングにおけるMPIを介した通信方法やプログラムの記述方法について述べるとともに、MPIライブラリの作成方法やMPIプログラムのコンパイルと実行方法について説明する。	9,10,11
II-10-5. プログラムの並列化手法	並列プログラミングを行う際の基本的な検討事項であるプログラムの並列化手法について説明する。計算部分や入出力における並列化のパターンや、ループを並列プログラムに分割する方法など、どのような部分が並列化できるかを解説する。	9,10,11
II-10-6. 並列プログラミングの応用例	並列プログラミングの様々な応用例を示す。応用例として、ブロック分割を利用した差分法、有限要素法の並列化、LU分解、ICCG法、マルチフロンタル法、SOR法、モンテカル法、個別要素法/分子動力学法といった分野への適用を示す。	9,10,11
II-10-7. PCクラスタの構築方法	一般的なPCクラスタとして代表的なシステムであるBeowulf型のPCクラスタを解説する。同クラスタを構成する方法を示し、各種の設定手順と並列プログラムのコンパイルおよび実行例を示す。またベンチマークの方法も紹介する。	12
II-10-8. SCoreの導入	国産のHPCクラスタ専用ミドルウェアSCoreの概要と特徴を紹介し、SCoreのアーキテクチャ、構成方法、インストール手順と利用方法について説明する。	13
II-10-9. HPCクラスタ向けユーティリティの利用	バッチ処理システム、システム監視ツール、並列処理ライブラリ、並列プロファイラ、並列プログラム向けデバッガ、並列プログラム開発環境、並列プログラムのベンチマークソフトウェアなど、HPCクラスタで効果的に活用できる各種のユーティリティソフトウェアの機能と特徴を説明する。	14
II-10-10. グリッドコンピューティング	グリッドコンピューティングの概念と各種のグリッドコンピューティングについて説明する。また米国、英国、欧州、日本におけるグリッドコンピューティングに関連する様々なプロジェクトを紹介する。	15

【学習ガイダンスの使い方】

1. 「習得ポイント」により、当該科目で習得することが期待される概念・知識の全体像を把握する。
2. 「シラバス」、「IT 知識体系との対応関係」、「OSS モデルカリキュラム固有知識」をもとに、必要に応じて、従来の IT 教育プログラム等との相違を把握した上で、具体的な講義計画を考案する。
3. 習得ポイント毎の「学習の要点」と「解説」を参考にして、講義で使用する教材等を準備する。

3. IT 知識体系との対応関係

「10. クラスタシステム構築に関する知識Ⅱ」と IT 知識体系との対応関係は以下の通り。

科目名	基本レベル（Ⅰ）							応用レベル（Ⅱ）							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10. クラスタシステム構築に関するスキル	<クラスタシステム概論、HA クラスタ(1)>	<HAクラスタ(2)>	<HAクラスタ(3)>	<コンピュータシミュレーション>	<並列プログラミング概論>	<並列プログラミング　実践(1)　マルチスレッドプログラミング>		<並列プログラミング実践(2) HPF (High Performance Fortran) と OpenMP>	<並列プログラミング　実践(3)　MPI (Message Passing Interface)>			<Beowulf PC クラスタの構築>	<SCore クラスタ>	<PC クラスタの周辺技術>	<グリッド・コンピューティング>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_10.pdf]

<IT 知識体系上の関連部分>

分野		科目名	1	2	3	4	5	6	7	8	9	10	11	12	13
組織間連事項と情報システム	1	IT-IAS 情報保証と情報セキュリティ	IT-IAS1. 基礎的な問題	IT-IAS2. 情報セキュリティの仕組み（対策）	IT-IAS3. 運用上の問題	IT-IAS4. ポリシー	IT-IAS5. 攻撃	IT-IAS6. 情報セキュリティ分野	IT-IAS7. フォレンジック（情報証拠）	IT-IAS8. 情報の状態	IT-IAS9. 情報セキュリティサービス	IT-IAS10. 脅威分析モデル	IT-IAS11. 脆弱性		
	2	IT-SP 社会的な観点とプロフェッショナルとしての課題	IT-SP1. プロフェッショナルとしてのコミュニケーション	IT-SP2. コンピュータの歴史	IT-SP3. コンピュータを取り巻く社会環境	IT-SP4. チームワーク	IT-SP5. 知的財産権	IT-SP6. コンピュータの法的問題	IT-SP7. 組織の中の IT	IT-SP8. プロフェッショナルとしての倫理的な問題と責任	IT-SP9. プライバシーと個人の自由				
応用技術	3	IT-IM 情報管理	IT-IM1. 情報管理の概念と基礎	IT-IM2. データベース問合わせ言語	IT-IM3. データアーキテクチャ	IT-IM4. データモデリングとデータベース設計	IT-IM5. データと情報の管理	IT-IM6. データベースの応用分野							
	4	IT-WS Web システムとその技術	IT-WS1. Web 技術	IT-WS2. 情報アーキテクチャ	IT-WS3. デジタルメディア	IT-WS4. Web 開発	IT-WS5. 脆弱性	IT-WS6. ソーシャルソフトウェア							
ソフトウェアの方法と技術	5	IT-PF プログラミング基礎	IT-PF1. 基本データ構造	IT-PF2. プログラミングの基本的構成要素	IT-PF3. オブジェクト指向プログラミング	IT-PF4. アルゴリズムと問題解決	IT-PF5. イベント駆動プログラミング	IT-PF6. 再帰							
	6	IT-IPT 技術を統合するためのプログラミング	IT-IPT1. システム間通信	IT-IPT2. データ割り当てと交換	IT-IPT3. 統合的コーディング	IT-IPT4. スクリプティング手法	IT-IPT5. ソフトウェアセキュリティの実現	IT-IPT6. 種々の問題	IT-IPT7. プログラミング言語の概要						
	7	CE-SME ソフトウェア工学	CE-SME0. 歴史と概要	CE-SME1. ソフトウェアプロセス	CE-SME2. ソフトウェアの要求と仕様	CE-SME3. ソフトウェアの設計	CE-SME4. ソフトウェアのテストと検証	CE-SME5. ソフトウェアの保守	CE-SME6. ソフトウェア開発・保守ツールと環境	CE-SME7. ソフトウェアプロジェクト管理	CE-SME8. 言語翻訳	CE-SME9. ソフトウェアのフォールトトレランス	CE-SME10. ソフトウェアの構成管理	CE-SME11. ソフトェアの標準化	
	8	IT-SIA システムインテグレーションとアーキテクチャ	IT-SIA1. 要求仕様	IT-SIA2. 調達/手配	IT-SIA3. インテグレーション	IT-SIA4. プロジェクト管理	IT-SIA5. テストと品質保証	IT-SIA6. 組織の特性	IT-SIA7. アーキテクチャ						
システム基盤	9	IT-NET ネットワーク	IT-NET1. ネットワークの基礎	IT-NET2. ルーティングとスイッチング	IT-NET3. 物理層	IT-NET4. セキュリティ	IT-NET5. アプリケーション分野	IT-NET6. ネットワーク管理							
	10	CE-NWK テレコミュニケーション	CE-NWK0. 歴史と概要	CE-NWK1. 通信ネットワークのアーキテクチャ	CE-NWK2. 通信ネットワークのプロトコル	CE-NWK3. LAN と WAN	CE-NWK4. クライアントサーバコンピューティング	CE-NWK5. データの セキュリティと整合性	CE-NWK6. ワイヤレスコンピューティングとモバイルコンピューティング	CE-NWK7. データ通信	CE-NWK8. 組込み機器向けネットワーク	CE-NWK9. 通信技術とネットワーク概要	CE-NWK10. 性能評価	CE-NWK11. ネットワーク管理	CE-NWK12. 圧縮と伸張
			CE-NWK13. クラスタシステム [10-1] [10-9, 10, 11]	CE-NWK14. インターネットアプリケーション	CE-NWK15. 次世代インターネット	CE-NWK16. 放送									
	11	IT-PT プラットフォーム技術	IT-PT1. オペレーティングシステム	IT-PT2. アーキテクチャと機構	IT-PT3. コンピュータインフラストラクチャ	IT-PT4. デプロイメントソフトウェア	IT-PT5. ファームウェア	IT-PT6. ハードウェア							
	12	CE-OPS オペレーティングシステム	CE-OPS0. 歴史と概要	CE-OPS1. 並行性	CE-OPS2. スケジューリングとディスパッチ	CE-OPS3. メモリ管理	CE-OPS4. セキュリティと保護	CE-OPS5. ファイル管理	CE-OPS6. リアルタイム OS	CE-OPS7. OS の概要	CE-OPS8. 設計の原則	CE-OPS9. デバイス管理	CE-OPS10. システム性能評価		
コンピュータアーキテクチャ	13	CE-CA0 コンピュータのアーキテクチャと構成	CE-CA00. 歴史と概要	CE-CA01. コンピュータアーキテクチャの基礎	CE-CA02. メモリシステムの構成とアーキテクチャ	CE-CA03. インタフェースと通信	CE-CA04. デバイスサブシステム	CE-CA05. CPU アーキテクチャ	CE-CA06. 性能・コスト評価	CE-CA07. 分散・並列処理 [10-5]	CE-CA08. コンピュータによる計算	CE-CA09. 性能向上			
複数領域にまたがるもの	14	IT-ITF IT 基礎	IT-ITF1. IT の一般的なテーマ	IT-ITF2. 組織の問題	IT-ITF3. IT の歴史	IT-ITF4. IT 分野（学科）とそれに関連のある分野（学科）	IT-ITF5. 応用領域	IT-ITF6. IT 分野における数学と統計学の活用							
	15	CE-ESY 組込みシステム	CE-ESY0. 歴史と概要	CE-ESY1. 低電力コンピューティング	CE-ESY2. 高信頼性システムの設計	CE-ESY3. 組込み用アーキテクチャ	CE-ESY4. 開発環境	CE-ESY5. ライフサイクル	CE-ESY6. 要件分析	CE-ESY7. 仕様定義	CE-ESY8. 構造設計	CE-ESY9. テスト	CE-ESY10. プロジェクト管理	CE-ESY11. 並行設計（ハードウェア、ソフトウェア	CE-ESY12. 実装
			CE-ESY13. リアルタイムシステム設計	CE-ESY14. 組込みマイクロコントローラ	CE-ESY15. 組込みプログラム	CE-ESY16. 設計手法	CE-ESY17. ツールによるサポート	CE-ESY18. ネットワーク型組込みシステム	CE-ESY19. インタフェースシステムと混合信号システム	CE-ESY20. センサ技術	CE-ESY21. デバイスドライバ	CE-ESY22. メンテナンス	CE-ESY23. 専門システム	CE-ESY24. 信頼性とフォールトトレランス	

4. OSS モデルカリキュラム固有の知識

OSS モデルカリキュラム固有の知識として、Linux 上での並列プログラミングと Linux 上で動作するクラスタシステムがある。並列プログラミング言語とグリッドコンピューティングに関する知識を Linux 上の OSS 実装を通して習得する。

科目名	第8回	第9回	第10回	第11回	第12回	第13回	第14回	第15回
10.クラスタシステム構築に関する知識Ⅱ	(1) HPF (High Performance Fortran)入門 (2) HPF プログラミング (3) OpenMP	(1) MPI (Message Passing Interface) (2) MPI によるプログラミング (3) プログラミングの並列化の方法 (4) 並列化の応用例			(1) Beowulf PC クラスタのコンポーネント (2) システム構築 (3) mpich2 のサンプルプログラムのコンパイル・実行 (4) ベンチマーク測定	(1) SCore の紹介 (2) SCore の優位性 (3) SCore のソフトウェアアーキテクチャ (4) ユーザ環境 (5) SCore のインストール (6) SCore のデモンストレーション	(1) バッチ処理システムの紹介 (2) システム監視の紹介 (3) 並列ライブラリの紹介 (4) 並列デバッカ、プロファイラの紹介 (5) 開発環境の紹介 (6) ベンチマークソフトウェアの紹介 (7) PC クラスタで使用可能な商用ソフト	(1) グリッド・コンピューティングの概要 (2) グリッド・コンピューティングの分類 (3) グリッド・コンピューティング関連プロジェクトの紹介 (4) The Globus Alliance と Globus Toolkit

（網掛け部分は IT 知識体系で学習できる知識を示し、それ以外は OSS モデルカリキュラム固有の知識を示している）

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	10. クラスタシステム構築に関する知識	応用
習得ポイント	II-10-1.　HPF (High Performance Fortran)の基礎	
対応するコースウェア	第 8 回 (並列プログラミング実践その 2 HPF (High Performance Fortran) と OpenMP)	

II-10-1. HPF (High Performance Fortran)の基礎

並列プログラミングを可能にする HPF (High Performance Fortran)を説明する。データを並列化するデータマッピング、処理を並列化する計算マッピングの概念を示し、簡単な HPF プログラミングについて解説する。

【学習の要点】

* HPF (High Performance Fortran)とは並列コンピューティングをサポートするように、Fotran90 を拡張して作られた言語である。
* HPF の特徴は、並列計算に必要なプロセッサ間の通信や同期の指示を、HPF コンパイラが自動生成することにあり、HPF には最初から並列計算を行うために必要な、データマッピング指示文、計算マッピング及び通信制御指示文、マッピング問い合わせ、配列演算ライブラリを含んでいる。
* データマッピングとは、並列化された複数のプロセッサが持つメモリ上にデータを分割し、割り当てることである。
* 計算マッピングとは、並列化された複数のプロセッサに計算処理を分割し、割り当てることである。

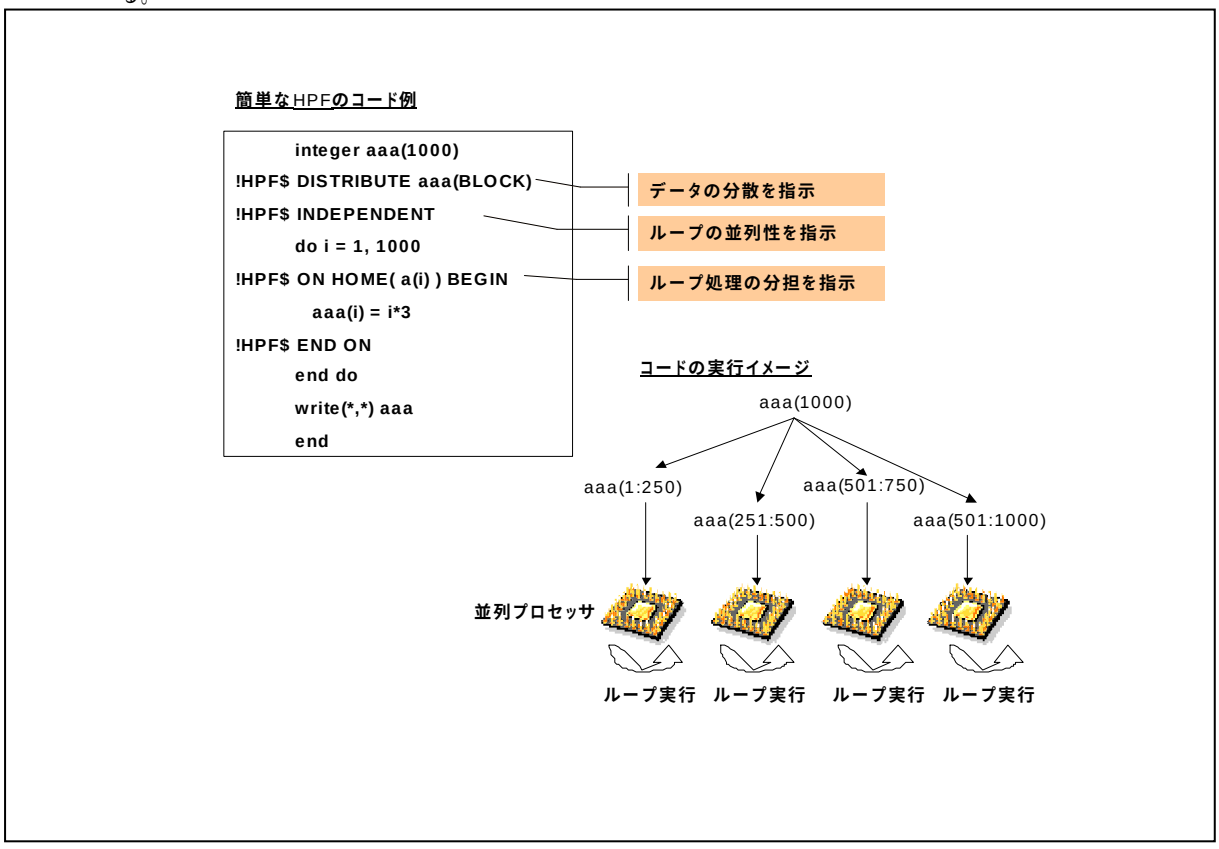


図 II-10-1.　HPF (High Performance Fortran)のコード例と実行イメージ

【解説】

1) HPF とは

* HPF (High Performance Fortran)とは並列コンピューティングをサポートするように、Fotran90 を拡張して作られた言語である。Rice 大学の Ken Kennedy 氏により組織された HPFF(HPF Forum, http://hpff.rice.edu/) にて策定された。
* HPF は MPI (Message Passing Interface) と異なり、並列計算に必要なプロセッサ間の通信や同期の指示を、HPF コンパイラが自動生成する。
* HPF はプログラマが指示文といわれる特殊構文をもちいて、通常の Fortran90 コードに並列化の指示を追記するという、発想から仕様が策定されている。プログラマは並列計算を行う部分に指示文を指定する。
* HPF では指示文は !HPF$ で始まる特殊な構文で記述する。
* HPF は並列計算を行うために必要な以下の要素を持っている。
  - データマッピング指示文
  - 計算マッピング及び通信制御指示文
  - マッピング問い合わせ
  - 配列演算ライブラリ

2) データマッピング

* データマッピングとは、並列化された複数のプロセッサが持つメモリ上にデータを分割し、割り当てることである。
* 注意することはデータを部分に分けてリモートプロセッサに割り当てるので、できる限り同じ処理に関連するデータは、同じプロセッサに割り当てる必要があるということである。

3) 計算マッピング

* 計算マッピングとは、並列化された複数のプロセッサに計算処理を分割し、割り当てることである。
* 注意することは、一つのプロセッサが処理を行うときは、自分のメモリにあるデータしか参照できないことである。そのために、もしもリモートのプロセッサのメモリ上にあるデータを参照したい場合は、データをリモートプロセッサから転送しなければならない。

4) オープンソース実装

* HPF のオープンソース実装として、ADAPTOR がある。
  (http://www.scai.fraunhofer.de/EP-CACHE/adaptor/)

5) 処理フロー

* 図の例では、4 つの並列プロセッサにデータマッピングと計算マッピングを行い、ループ処理を実行させている。

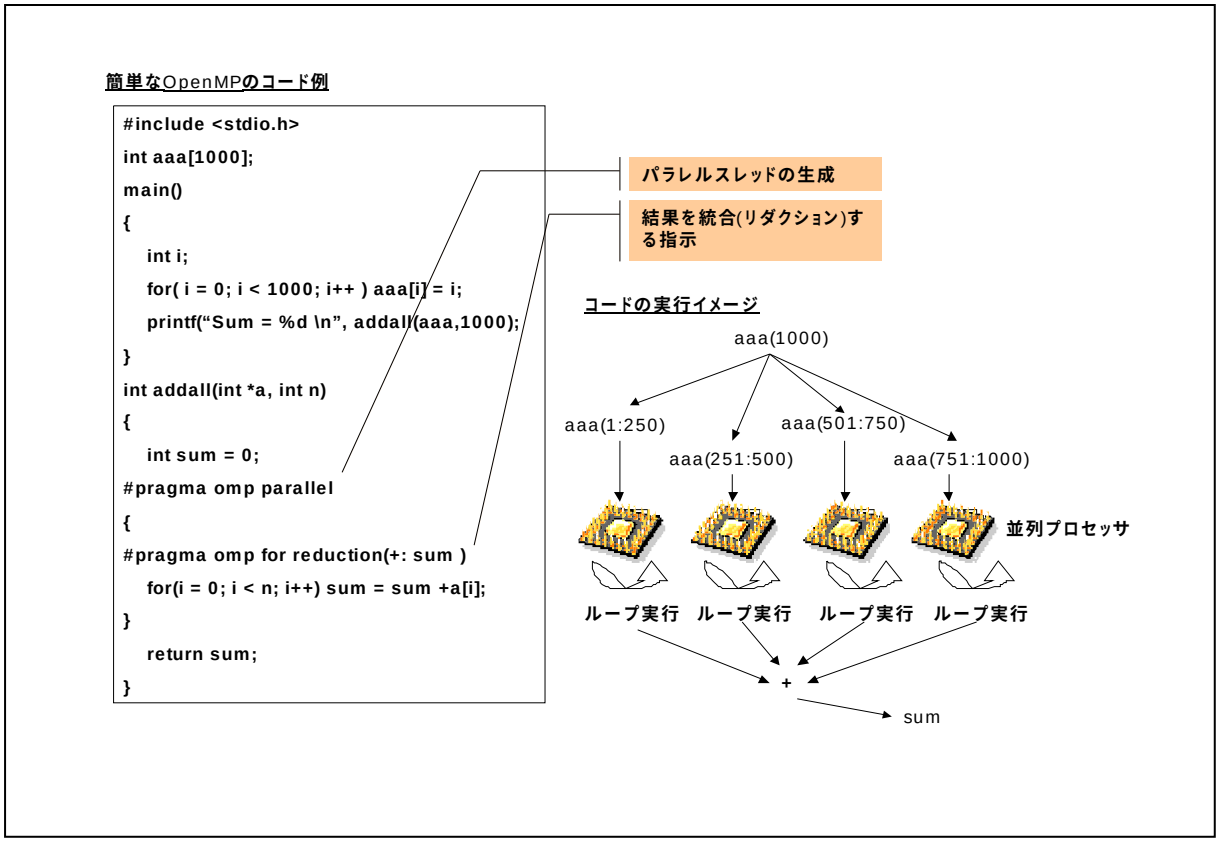
スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	10. クラスタシステム構築に関する知識	応用
習得ポイント	Ⅲ-10-2. OpenMP による並列プログラミング	
対応するコースウェア	第 8 回（並列プログラミング実践その 2 HPF (High Performance Fortran) と OpenMP)	

Ⅲ-10-2. OpenMP による並列プログラミング

並列プログラミング向けのライブラリである OpenMP の概要とその利用方法を解説する。並列実行領域の確保、ワークシェアリング、データ環境、同期構文、実行時間数など各種の API を紹介し、簡単なサンプルによる具体的な利用法を示す。

【学習の要点】

* OpenMP は共有メモリマルチプロセッサ上のマルチスレッドプログラミングのための API である。C/C++、Fortran そして Java に対してコンパイラ実装がある。
* OpenMP は既存の逐次実行プログラムに OpenMP 指示文を加えることで並列プログラミングを行う。
* OpenMP 指示文には、並行実行領域を確保する指示文、ワークシェアリング指示文、データ環境指示文、同期構文などがある。指示文のほかには、実行時ライブラリ関数がある。



【解説】

図 Ⅲ-10-2.　OpenMP によるコード例と実行イメージ

1) OpenMP とは

* OpenMP は共有メモリマルチプロセッサ上のマルチスレッドプログラミングのための API である。
* C/C++、Fortran そして Java に対してコンパイラ実装があり、その中でオープンソースソフトウェアとしては GCC(4.1 より上のバージョン)や Omni OpenMP Compiler 等がある。
* OpenMP は新しい言語ではなく、既存の逐次言語に OpenMP 指示文を加えることで並列プログラミングを行う。
* OpenMP 指示文として、C 言語では#Pragma 文や Fortran ではコメント行$!を利用する。

2) OpenMP の特徴

* 逐次プログラムをベースに並列プログラムを段階的につくることができる。
* 指示文を利用するだけなので、スレッドライブラリを利用するスレッドプログラミングよりも簡単である。
* 指示文はコメントとして扱えば逐次プログラムとコードベースを共有できるので、逐次プログラムと並列プログラムとうい二つのコードを別管理しなくてよく、ソースコード管理については比較的コストが低く済む。
* MPI (「II-10-3 MPI の基礎」参照) に比べると実行速度が遅い。

3) OpenMP の規約

* 並行実行領域を確保する指示文
  - parallel 文(#pragma omp parallel)
* ワークシェアリング指示
  - for 指示文(#pragma omp for sahred/private/firstprivate/lastprivate/reduction)
  - single 指示文(#pragma omp single)
  - section 指示文(#pragma omp sections)
* データ環境指示文
  - threadprivate 指示文(#pragma omp threadprivate (list)new-line)
* 同期構文
  - master 指示文(#pragma omp master new-line)
  - critical 指示文(#pragma omp critical new-line)
  - barrier 指示文(#pragma omp barriernew-line)
  - atomic 指示文(#pragma omp atomicnew-line)
  - flush 指示文(#pragma omp flush new-line)
  - orderd 指示文(#pragma omp orderd new-line)
* 実行時ライブラリ関数
  - omp_get_num_threads：並列処理を実行しているチームのスレッド数を返す

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	10. クラスタシステム構築に関する知識	応用
習得ポイント	II-10-3.　MPI の基礎	
対応する コースウェア	第 9, 10, 11 回 (並列プログラミング実践その3)	

II-10-3. MPI の基礎

HPC クラスタにおける並列プログラミングで一般的に利用される技術である MPI (Message Passing Interface)の概要とプログラミングモデル、基本機能の概要と、MPI の先端的な特徴について解説する。

【学習の要点】

* MPI(Message Passing Interface)は並列プログラミングのための通信ライブラリである。
* MPI のプログラミングモデルは SPMD(Simgle Program Multiple Data)と呼ばれるもので、同一プログラムが複数の並列ノードにロードされ、プロセスとして実行される。
* ベンダー間の共通規格であり、ポータビリティが高い。
* オーバーヘッドが少ないために、実行パフォーマンスがよい。
* MPI ライブラリ分類として、コミュニケータ、一対一通信、集団通信、派生データ型、プロセス・トポロジー、環境管理の各種類がある。

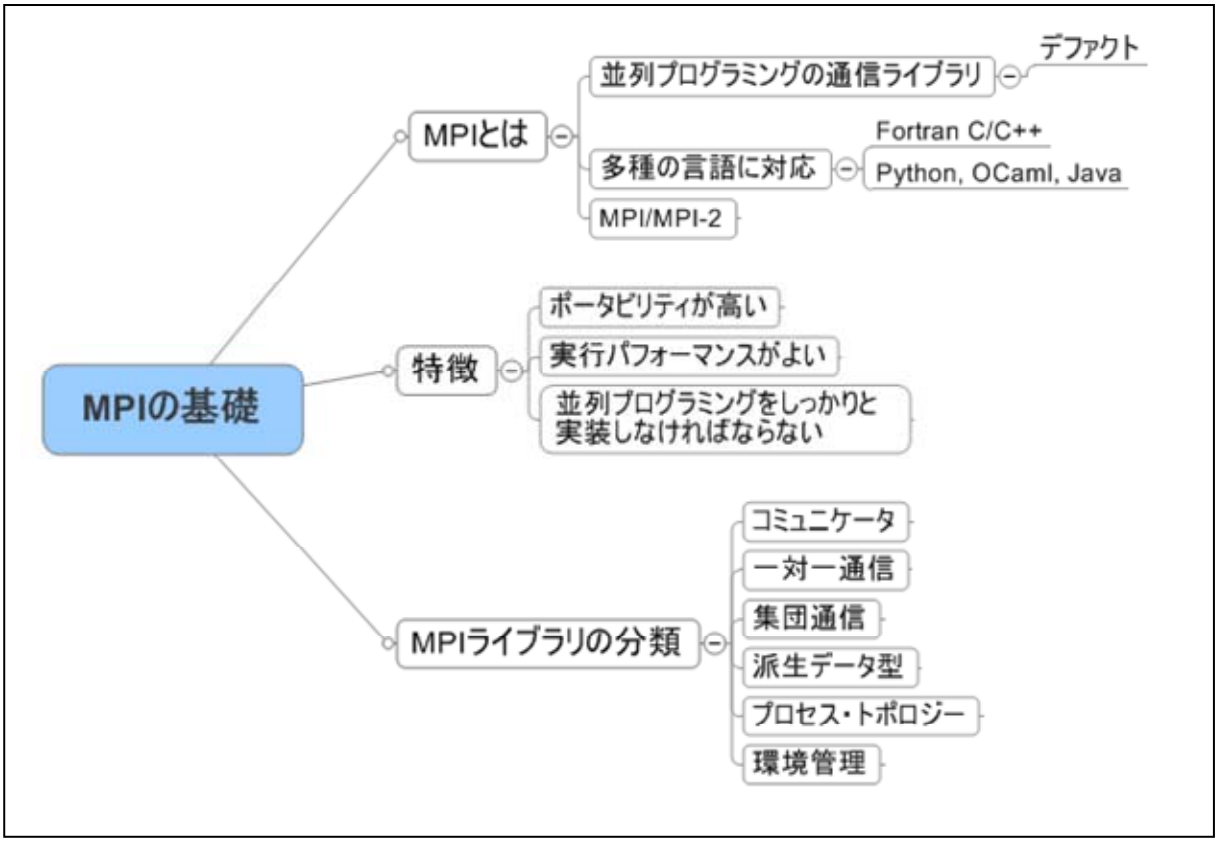


図 II-10-3.　MPI の概要

【解説】

1) MPI とは

* MPI(Message Passing Interface)は言語に独立な並列プログラミングのための通信ライブラリである。MPI はハイパフォーマンスコンピューティングの現在のデファクトスタンダードとなっている。
* 中心的な実装としては C/C++または Fortran 77 がある。最近では、Python、 Ocaml、 Java 等の実装も利用できる。
* MPI の規格は MPI と拡張仕様の MPI-2 があり、並列プログラミングのための豊富な通信ライブラリ環境を提供している。
* MPI のプログラミングモデルは SPMD(Simgle Program Multiple Data)と呼ばれるもので、同一プログラムが複数の並列ノードにロードされ、プロセスとして実行される。各ノードにおいては一斉に各プロセスが実行されるためにノード間に主従関係がない。

2) MPI の特徴

* 基本的には言語非依存の仕様のために、ポータビリティが高い。並列プログラミング専用のため、関数呼び出しやノード間通信のオーバーヘッドが少ないために、実行パフォーマンスがよい。
* プログラミングは、逐次言語の拡張というわけにはいかない。MPI ライブラリを利用する上で並列プログラミングをしっかり意識してコーディングをする必要がある。

3) MPI ライブラリの分類（主に MPI-1)

* コミュニケータ
  MPI でプロセス通信を行うオブジェクトをコミュニケータという。このコミュニケータを扱うライブラリがコミュニケータライブラリである。
* 一対一通信
  一対一通信は二つのプロセスの間で行われる通信を取り扱うライブラリである。
* 集団通信
  集団通信は、一回のライブラリコールにより複数のプロセスがメッセージを一斉に交換するような通信を取り扱うライブラリである。MPI ライブラリでは最も利用頻度が高い。
* 派生データ型
  派生データ型とは、メモリ上でとびとびにはいっている配列上のデータを、やり取りするデータ型である。MPI の通信ライブラリは、連続したデータしか通信として取り扱うことができない。そのために、とびとびのデータは一度連続した配列にバッファリングしてから、MPI の通信に乗せる必要があった。しかし、派生データ型を用いればバッファリングなしに、コミュニケータ同士で直接データを通信することができる。
* プロセス･トポロジー
  MPI において、プロセスのグループは通常線形順序集合として取り扱われる。このプロセス･トポロジーライブラリを利用すれば、多次元グリッドのような、より複雑なトポロジー構造に対応づけることができる。
* 環境管理
  環境管理ライブラリは、並列プログラミングの環境を設定･管理･変更するライブラリである。MPI プログラミングを行う際に必ず MPI_Init( )と MPI_Finalize( )が呼ばれるが、この二つの関数にはさまれた部分が、MPI の並列プログラミング環境になる。

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	10. クラスタシステム構築に関する知識	応用
習得ポイント	II-10-4.　MPI による並列プログラミング	
対応する コースウェア	第 9、 10、 11 回 (並列プログラミング実践その 3)	

II-10-4. MPI による並列プログラミング

MPI を用いた具体的な並列プログラミングの例を示す。並列プログラミングにおける MPI を介した通信方法やプログラムの記述方法について述べるとともに、MPI ライブラリの作成方法や MPI プログラムのコンパイルと実行方法について説明する。

【学習の要点】

* MPI ライブラリを利用するには、MPI ライブラリの生成とインストールを行う必要がある。
* MPI ライブラリを利用したプログラムをコンパイルするには適切なコンパイラドライバをシステムにインストールすれば、通常の C ライブラリと同様にしてコンパイルを行うことができる。
* コンパイルが完了した実行ファイルの実行方法は環境において異なるが、基本的には mpirun コマンドを用いる。
* MPI を利用したプログラムの記述法は、プログラマはそれが並列プログラムであること明確に意識しながらコーディングを行う必要がある。

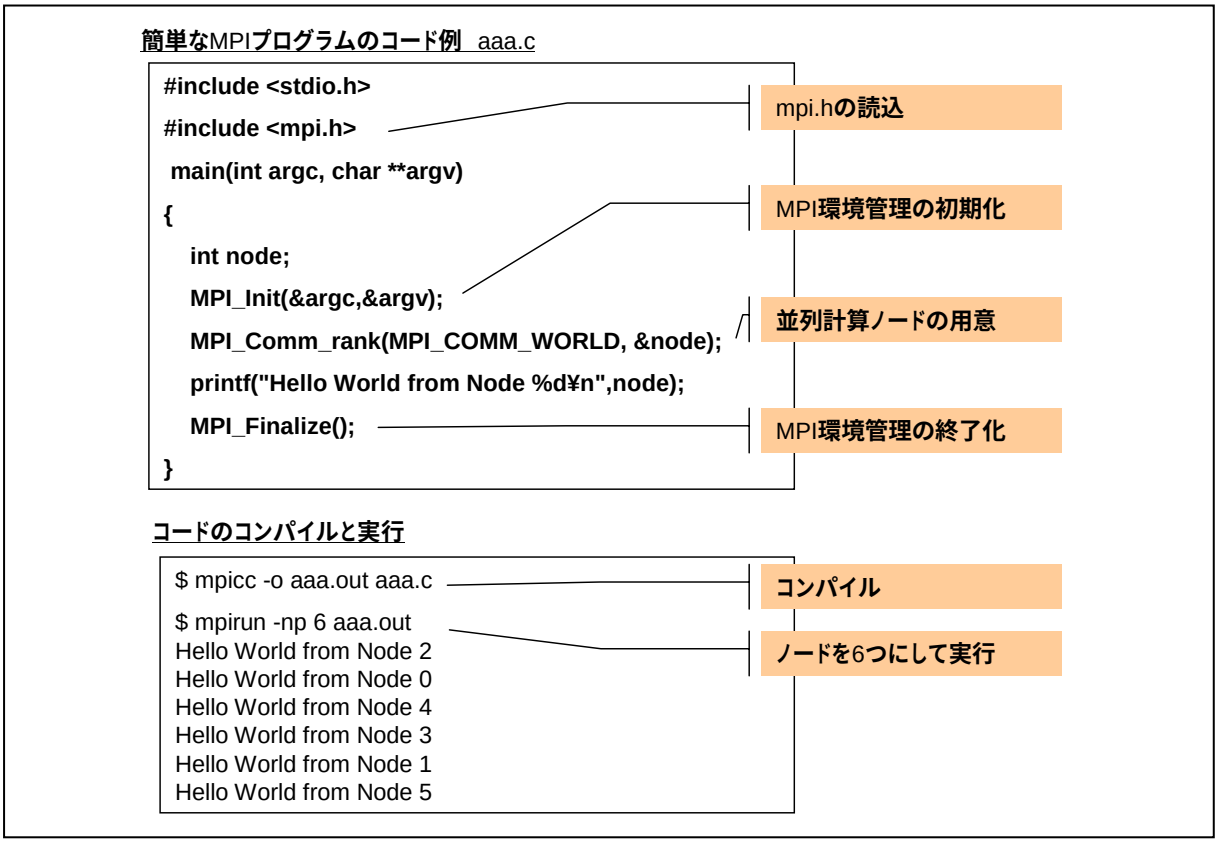


図 II-10-4.　MPI による並列プログラミング

【解説】

1) MPI ライブラリの準備

* MPI ライブラリの生成とインストールは、通常、以下の手順で実施する。
  - 環境変数の設定
  - configure シェルの実行
  - make コマンドによるメイク
  - make install によるインストール

2) MPI プログラムのコンパイル

* 適切なコンパイラドライバをシステムにインストールすれば、通常の C ライブラリと同様に、MPI ライブラリを利用したプログラムをコンパイルすることができる。
  例えば、SCore 環境において C 言語をコンパイルするには mpicc ドライバがあり、次のようにコンパイルを行う:
  - mpicc -o aaa.out aaa.c

3) MPI プログラムの実行

* コンパイルが完了した実効ファイルの実効方法は環境において異なるが、基本的には mpirun コマンドを用いる。
* mpirun コマンドはノード数を指定する必要がある。
  - mpirun -np 6 aaa.out

4) MPI プログラムの記述方法

* MPI を利用したプログラムの記述法は、プログラマはそれが並列プログラムであること明確に意識し、各ノードにどのメモリが割り当てられ、どの処理が実行されるかを常に理解しながらコーディングを行う必要がある。
* OpenMP や HPF(High Performance Fortran)の様に既存のプログラムにディレクティブ (指示文) を挿入するだけで並列プログラムができてしまうようなものではない。
* MPI ライブラリを利用する初歩は以下の通りである。
  - mpi.h を読み込む
  - 環境管理の MPI_Init( )と MPI_Finalize( )で並列化部分を囲む
  - コミュニケータライブラリや集団通信ライブラリなどの MPI プログラムを利用して、並列化プログラミングを記述する。

5) MPI のオープンソース実装

* MPI のオープンソース実装として有名なのは MPICH である。MPICH は MPI-1 も MPI-2 も両方とも実装対応している。
  http://www.mcs.anl.gov/research/projects/mpich2/

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	10. クラスタシステム構築に関する知識	応用
習得ポイント	II-10-5. プログラムの並列化手法	
対応するコースウェア	第 9、 10、 11 回 (並列プログラミング実践その 3)	

II-10-5. プログラムの並列化手法

並列プログラミングを行う際の基本的な検討事項であるプログラムの並列化手法について説明する。計算部分や入出力における並列化や、ループを並列プログラムに分割する方法などを概説し、どのような部分が並列化できるかを解説する。

【学習の要点】

* プログラムの並列化で一番大切なのは並列性を確保することである。
* 回帰参照がある場合は、原則として並列化を行うことができない。
* 計算部分を並列化するポイントは、いくつかある。例:計算が集中しているところ、データを効率よく分散できるところ、階層されている計算の上位部分。
* プログラムを並列化した場合、データの入出力についても並列化を検討しなければならない場合がある。

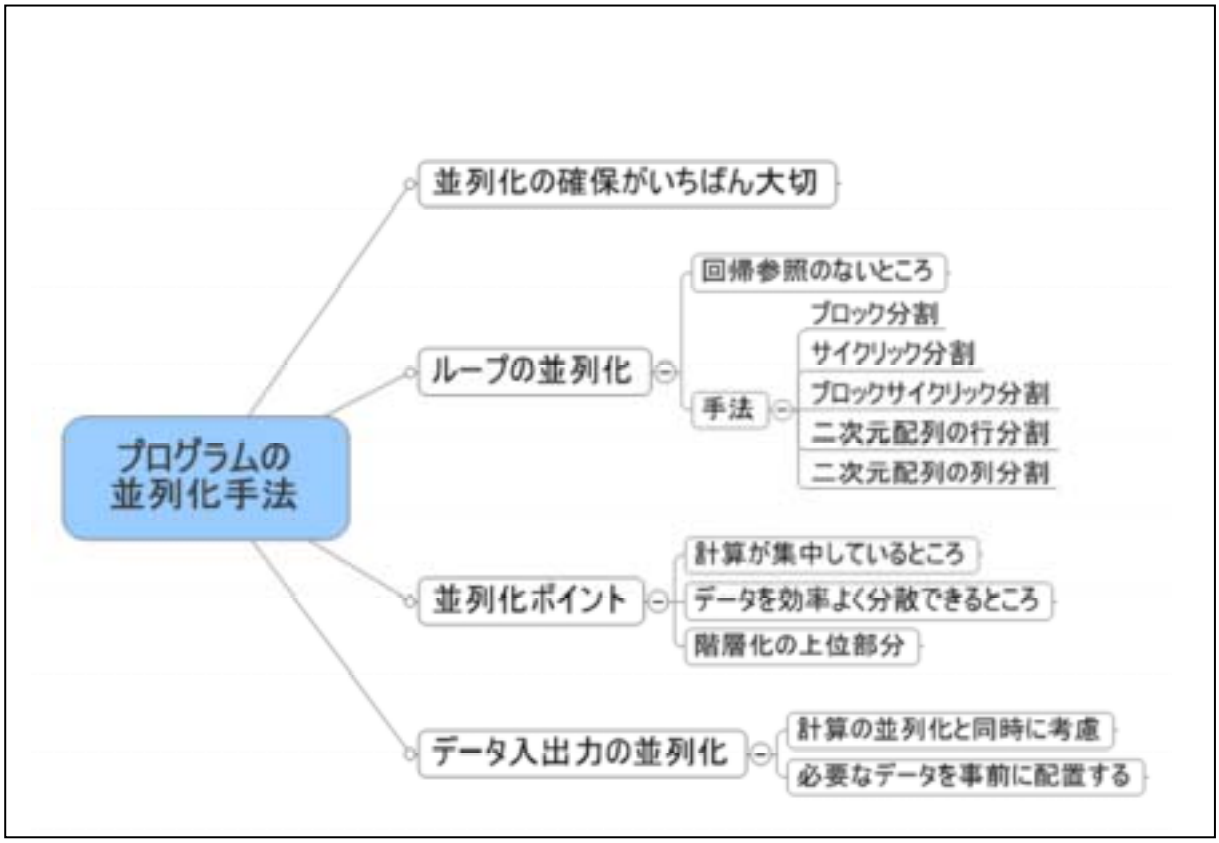


図 II-10-5. プログラムの並列化手法のポイント

【解説】

1) 並列化の原理

* プログラムの並列化を行うときに一番大切なのは並列性を確保するということである。並列性とはプログラムブロックが独立に実行できる性質である。

2) ループの並列化

ループを並列化するときに気をつけるのは、回帰参照があるかどうかを見定めることである。回帰参照がある場合は、原則として並列化を行うことができない。

回帰参照とは：配列がループを回るごとに自分自身の結果を参照しているケースで、以下のようなコード(例：C 言語) を回帰参照という。

```
for( i = 0; i < n; i++ ) {
    a[i+1] = a[i] + 3;
}
```

* 並列計算機を利用する場合、多重ループ、ライブラリコールを含む for ループ、入出力を含む for ループ、独立された処理ブロックのいずれも並列化を行うことが原理的にできる。ただし、それぞれの部分に必ず並列性がなければいけない。
* 並列化手法の種類
  - ブロック分割
  - サイクリック分割
  - ブロック・サイクリック分割
  - 二次元配列の行方向分割
  - 二次元配列の列方向分割

3) 計算部分を並列化するポイント

* 計算が集中しているところを並列化する。
* データを効率よく分散する。
* 階層化されている計算の上位部分を並列化する。

4) データの入出力の並列化

* プログラムを並列化した場合、データの入出力についても並列化を検討する場合がある。計算やループを並列化したとしても、入出力でボトルネックとなるからである。
* 入出力の並列化のポイントは、各ノードが処理を実行する前に、それに必要なデータを配置しておくというものである。

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	10. クラスタシステム構築に関する知識	応用
習得ポイント	II-10-6. 並列プログラミングの応用例	
対応するコースウェア	第 9、 10、 11 回 (並列プログラミング実践その 3)	

II-10-6. 並列プログラミングの応用例

並列プログラミングの様々な応用例を示す。応用例として、ブロック分割を利用した差分法、有限要素法の並列化、LU 分解、ICCG 法、マルチフロンタル法、SOR 法、モンテカルロ法、個別要素法/分子動力学法といった分野への適用を示す。

【学習の要点】

* 並列化プログラミングには様々は応用例がある。差分法、有限要素法、LU 分解、ICCG 法、SOR 法、モンテカルロ法、個別要素法/分子動力学法といった分野である。
* それぞれの手法を並列化するには、個別に対応した手法が必要で、ぞれぞれの手法に対しての深い理解と、並列化技術の熟練が必要とされる。

並列プログラミングの応用例

手法		各手法に対する並列プログラミングのポイント
差分法		ブロック分割を工夫する
有限要素法	単体法・陽解法	ループを並列化 接点を並列化 要素接点間のループを並列化
	陰解法	個別問題ごとの対応が必要
LU分解		ステップごとの行列サイズを変更 サイクリック分解
ICCG法		パイプライン法を利用 パラレル・フロック・オーダリング法を利用
SOR法		計算順序の変更
モンテカルロ法		乱数発生の並列化の工夫 (各ノードで乱数を分散させる、ノードごとに独自乱数を発生させる)
個別要素法 分子動力学法		回帰参照のない部分を並列化

図 II-10-6. 並列プログラミングの応用例

【解説】

1) 並列化プログラミングの応用例

* 並列化プログラミングには様々は応用例がある。差分法、有限要素法、LU 分解、ICCG 法、SOR 法、モンテカルロ法、個別要素法/分子動力学法といった分野である。
* それぞれの手法への並列化の一般的ルールは、回帰参照がない部分を並列化する、というものである。ただし、実際には具体的問題についての個別対応が多いために、様々な工夫が必要とされる。

2) 各種法への並列化の適用

* 差分法
  並列化プログラミングを適用するには、計算する配列をノードに分割する方法（ブロック分割）を工夫する必要がある。ノード間のデータの通信量が少なく済むような分割が、効率の良い分割法である。
* 有限要素法
  並列化プログラミングを適用するためには、単体法と陰解法、そして陽解法という手法ごとに工夫して並列化の分割を行う必要がある。単体法や陽解法については、要素のループ、節点のループ、要素-節点間のループを並列化する。陰解法については、一般的に難しく、個別事情に基づいた専門知識が求められる。
* LU 分解
  LU 分解並列化するには、LU 分解のステップごとの計算領域のサイズ変更を考慮した、行列の並列化分解が必要である。一般的に、行列を単純にブロックごとに分割する方法よりも、ノードごとの分解領域を小さくサイクリックにした、サイクリック分解が優位である。
* ICCG 法
  ICCG 法を並列化するには、依存するループを一続きで先に計算しその後並列化する、パイプライン法、または二次元ブロックに分割した後にパイプライン法を利用する、パラレル・ブロック・オーダリング法をもちいる。
* SOR 法
  並列化するには、計算順序を変更して並列性を持たせることで行う。その手法は多岐に渡り、あまりに専門性が高く、問題ごとに個別の対応が必要になる。
* モンテカルロ法
  並列化するには、乱数の発生を並列化する方法を工夫する必要がある。具体的には一つの乱数を各ノードに分散させる方法と、各ノードで独自に乱数を発生させる方法、そしてそれらを組み合わせる方法がある。
* 個別要素法/分子動力学法
  ループのみが計算の集中する部分になるので、比較的並列化に向いている。回帰参照がない部分を並列化するのがポイントである。

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	10. クラスタシステム構築に関する知識	応用
習得ポイント	II-10-7.　PC クラスタの構築方法	
対応する コースウェア	第 12 回 (Beowulf PC クラスタの構築)	

II-10-7. PC クラスタの構築方法

一般的な PC クラスタとして代表的なシステムである Beowulf 型の PC クラスタを解説する。同クラスタを構成する方法を示し、各種の設定手順と並列プログラムのコンパイルおよび実行例を示す。またベンチマークの方法も紹介する。

【学習の要点】

* Beowulf とは Linux マシンをノードとする分散メモリ型の PC クラスタシステムである。
* クラスタを構成する各 PC は Ethernet を介して接続され、NFS 経由で/home を共有し、rsh で通信を行う。このようにオープンソースプログラムと、安価な PC を利用できるために、非常に安くスーパーコンピュータと同等の計算環境を手に入れることができる。

  Beowulf における並列プログラミングは MPI を利用することが多い。
* Beowulf は、ファイルサーバ、マスタ演算ノード、スレーブ演算ノードから構成される。

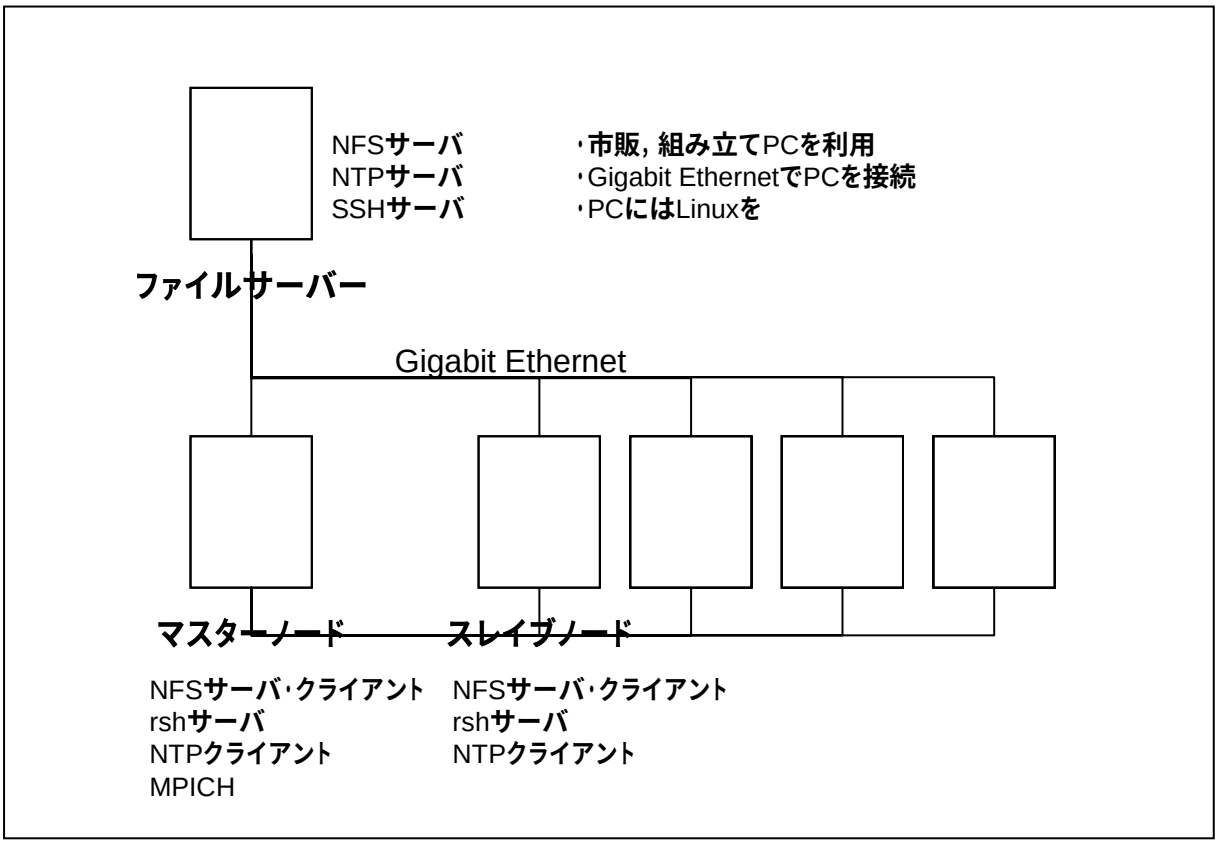


図 II-10-7.　PC クラスタの構築方法

【解説】

1) Beowulf PC クラスタとは

* Beowulf とは Linux マシンをノードとする分散メモリ型の PC クラスタシステムである。各 PC は Ethernet を介して接続され、NFS 経由で/home を共有し、rsh で通信を行う。
* 非常に安価にスーパーコンピュータと同等の演算能力を持つクラスタを構築することができる。
* Beowulf は、以下のノード構成となっている。
  - ファイルサーバ
  - マスタ演算ノード
  - スレーブ演算ノード
* ファイルサーバは以下のサーバの設定が必要である。
  - NFS サーバ
  - NTP サーバ
  - SSH サーバ
* マスタ演算ノードは、以下のサーバおよびクライアントの設定が必要である。
  - NFS サーバ、クライアント
  - rsh サーバ
  - NTP クライアント
  - MPICH
* スレーブ演算ノードは、以下のサーバおよびクライアントの設定が必要である。
  - NFS クライアント
  - rsh サーバ
  - NTP クライアント

2) Beowulf におけるコンパイル、実行

* Beowulf における並列プログラミングは MPI を利用することが多い。たとえば、オープンソースの MPICH を利用すると、MPI 環境を非常に安価に構築することができる。
* Beowulf における並列プログラミングのコンパイル･実行例については、「MPI における並列プログラミング」と同じである。

3) PC クラスタのベンチマーク

* PC クラスタのベンチマークには以下のものがよく使われる。中でも姫野ベンチは短時間で実速度を求めることが出来るので、国内ではよく使われる傾向がある。
  - 姫野ベンチ
    http://w3cic.riken.go.gjp/HPC/HimenoBMT
  - ScaLAPACK
    http://www.netlib.org/scalapack/
  - NAS Parallel Benchmarks
    http://www.nas.nasa.gov/Softwrare/NPB/

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	10. クラスタシステム構築に関する知識	応用
習得ポイント	II-10-8. SCore の導入	
対応する コースウェア	第 13 回 (SCore クラスタ)	

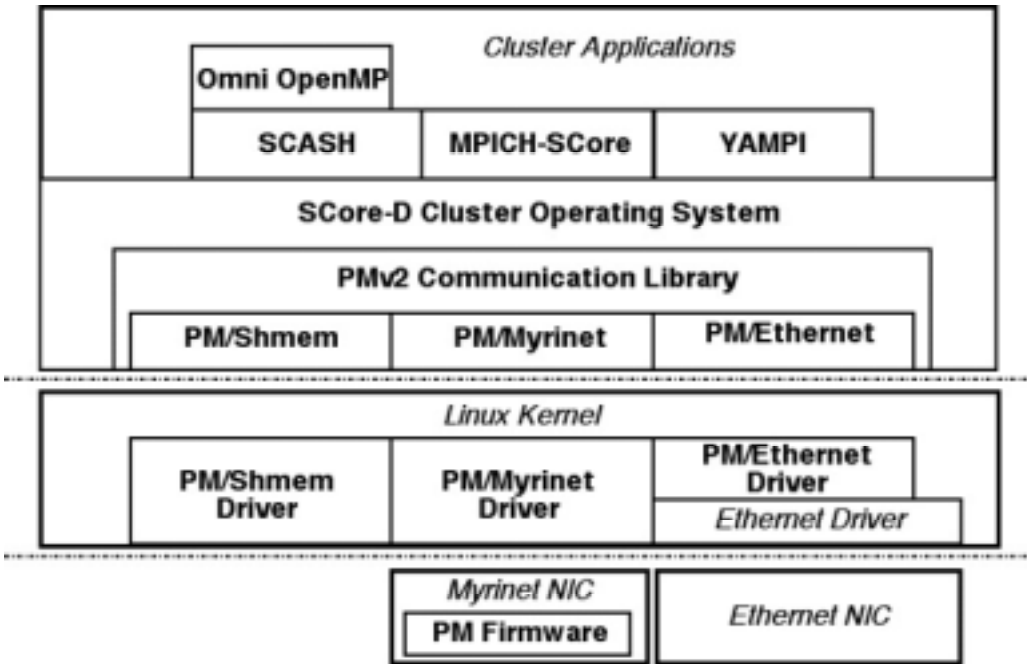
II-10-8. SCore の導入

国産の HPC クラスタ専用ミドルウェア SCore の概要と特徴を紹介し、SCore のアーキテクチャ、構成方法、インストールと実行について概説する。

【学習の要点】

* SCore は国産 HPC 専用ミドルウェアであり、Linux 上にクラスタ計算機用超並列プログラム実行環境を実現する。
* SCore の先端的特長として、TCP/IP を使用しない高性能かつ高信頼な通信と、高度なネットワーク上の分散リソース管理がある。
* インストールは、クラスタ候補の計算機群に NIS を設定した後に、RPM またはソースビルドにより行う。

SCore**のソフトウェアアーキテクチャ**



PC Cluster Consortium**のサイトより引用**
http://www.pccluster.org/

図 II-10-8. SCore のアーキテクチャ

【解説】

1) SCore とは

* 新情報処理開発機構(Real World Computing Partnership、 RWCP)にて開発された Linux 用クラスタ計算機用超並列プログラム実行環境のことである。現在では RWCP は解散され、その受け皿となった PC クラスタコンソーシアム(PCCC)が開発/普及活動を行っている。
* SCore は国産 HPC 専用ミドルウェアで、以下の構成からなっている。
  - 低レベル通信ライブラリ(PMv2 Communication Library)
    クラスタ コンピューティングにおいて多くの種類のネットワークや共有メモリに同一の方法でアクセスできるように設計されている。
  - クラスタ間のリソースを管理するオペレーティングシステム(SCore-D)
    種々の PM ネットワークデバイスは、SCore-D によって管理され利用される。SMP クラスタや異種混在クラスタも単一種クラスタと同様にサポートされる。SCore-D は、柔軟な並列ジョブスケジューリング機構を提供する。ユーザの並列ジョブは、同時にかつ効率的に時間と空間が共有される。
  - 分散共有メモリ(SCASH)
    PMv2 を用いたソフトウェア DSM (Distributed Shared Memory) システム
  - MPI / OpenMP コンパイラ(Omni OpenMP コンパイラ、MPC++)
  - デバッガ
  - 監視ツール

2) SCore の特徴

Beawulf クラスタと同様、市販の PC に Linux をインストールして、Ethernet を通じてクラスタを構成するので、ハードウェア、ソフトウェア共に安価な PC クラスタを構築することができる。

* SCore は単一システムイメージなので、ユーザがクラスタの細かな情報を知る必要がない。
* SCore はメッセージパッシングパラダイムだけでなく、共有メモリ並列プログラミングパラダイムや、マルチスレッド並列プログラミングパラダイム等の多重プログラミングパラダイムに対応している。
* SCore の先端的特長は、PMv2 と SCore-D にあり、TCP/IP を使用しない高性能かつ高信頼な多重ネットワーク通信と、高度なネットワーク上の分散リソース管理（リアルタイムプロセスモニタ、デッドロックの探知）に定評がある。
* プリエンティブチェックポイントを採用しているので、フォールトトレランス性に優れている。
* SCore-D はプロセッサリソースの多重処理を行うために、柔軟なジョブスケジューリングを実現できる。
* DNA 解析など、比較的データ数が少なく計算量が多い処理において使用される事が多い。

3) SCore のインストールと実行

* インストールは、クラスタ候補の計算機群に NIS を設定した後に、RPM またはソースビルドにより行え、比較的簡単に設定ができる。
* 実行は MPI 環境、または OpenMP 環境で行う。

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	10. クラスタシステム構築に関する知識	応用
習得ポイント	II-10-9. HPC クラスタ向けユーティリティの利用	
対応するコースウェア	第 14 回 (PC クラスタの周辺技術)	

II-10-9. HPC クラスタ向けユーティリティの利用

バッチ処理システム、システム監視ツール、並列処理ライブラリ、並列プロファイラ、並列プログラム向けデバッガ、並列プログラム開発環境、並列プログラムのベンチマークソフトウェアなど、HPC クラスタで効果的に活用できる各種のユーティリティソフトウェアの機能と特徴を説明する。

【学習の要点】

* HPC クラスタを利用するためには各種のユーティリティを利用すると有効である。
* HPC のユーティリティとしては、バッチ処理システム、システム監視、並列ライブラリ、並列デバッガ・プロファイラ、開発環境、ベンチマークソフトウェアなどがある。

HPCクラスタ向けユーティリティ

ユーティリティ種類	
バッチ処理	OpenPBS TORQUE GridEngine PBS LSF
システム監視	Ganglia
並列ライブラリ	ScaLAPACK NAG Parallel Library
並列デバガ、プロファイラ	Intel Trace Coolector / Analyzer TotalView
開発環境	PTP(Parallel Tools Platform)
ベンチマークソフトウェア	**姫野ベンチ** ScaLPACK NAS Parallel Benchmarks SKaMPI Intel MPI Benchmarks

図 II-10-9. HPC クラスタ向けの主なユーティリティ

【解説】

1) バッチ処理システム

* OpenPBS（オープンソース）
  http://www.pbsgridworks.com/PBSTemp1.3.aspx?top_nav_name=Products&item_name=OpenPBS&top_nav_str=1&AspxAutoDetectCookieSupport=1
* TORQUE
  http://www.clusterresources.com/
* GridEngine
  http://gridengine.sunsource.net/
* PBS
  http://www.pbsgridworks.com/Default.aspx
* LSF
  http://www.platform.com/Products/platform-lsf-family/platform-lsf/product

2) システム監視

* Ganglia　http://ganglia.info/

3) 並列ライブラリ

* ScaLAPACK　http://www.netlib.org/scalapack/
* NAG Parallel Library　http://www.nag.co.uk/numeric/fd/fddescription.asp

4) 並列デバガ、プロファイラ

* Intel Trace Collector/Analyzer
  http://www.intel.com/cd/software/products/asmo-na/eng/306321.htm
* TotalView　http://www.totalviewtech.com/index.htm

5) 開発環境

* PTP (Parallel Tools Platform) : Eclipse のプラグイン　http://www.eclipse.org/ptp/

6) ベンチマークソフトウェア

* 姫野ベンチ　http://w3cic.riken.go.gjp/HPC/HimenoBMT
* ScaLAPACK　http://www.netlib.org/scalapack/
* NAS Parallel Benchmarks　http://www.nas.nasa.gov/Softwrare/NPB/
* SKaMPI　http://liinwww.ira.uka.de/~skampi/
* Intel MPI Benchmarks
  http://www3.intel.com/cd/software/products/asmo-na/eng/219848.htm

スキル区分	OSS モデルカリキュラムの科目	レベル
システム分野	10. クラスタシステム構築に関する知識	応用
習得ポイント	II-10-10. グリッドコンピューティング	
対応するコースウェア	第 15 回 (グリッドコンピューティング)	

II-10-10. グリッドコンピューティング

グリッドコンピューティングの概念と各種のグリッドコンピューティングについて説明する。また米国、英国、欧州、アジア、そして日本におけるグリッドコンピューティングに関連する様々なプロジェクトを紹介する。

【学習の要点】

* グリッドコンピューティングは、並列分散処理の特殊な形態の一つである。コンピュータクラスタに比べて、より粗に結合され、地理的に離れた多様なコンピュータ同士が協調して計算するアーキテクチャである。
* グリッドコンピューティングの大ざっぱな分類としては、プロセッシンググリッド、データグリッド、ビジネスグリッドとなっている。
* グリッドコンピューティングはビジネスでの適用はもちろんのこと、重要な国家プロジェクトとして、世界各国で競争が行われている分野である。

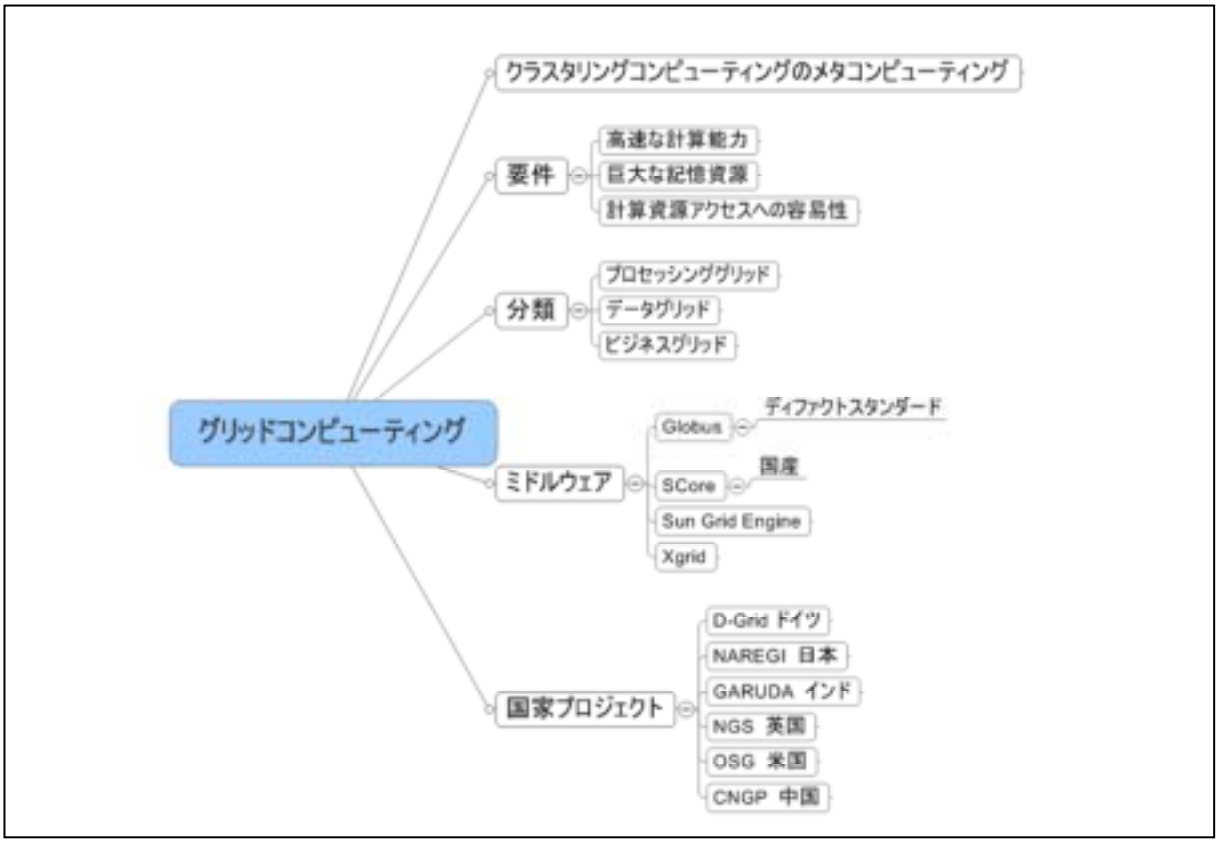


図 II-10-10. グリッドコンピューティングのポイント

【解説】

1) グリッドコンピューティングとは

* クラスタリングコンピューティング（並列コンピューティングそして分散情報システム）とは、複数の計算機を組み合わせて、計算処理の向上を示すものである。これに対してグリッドコンピューティングとは、違ったクラスタリングコンピュータ同士さえも接続して、巨大なクラスタリング・クラスタリングコンピューティングを構築する仕組みである。
* グリッドコンピューティングは OS や物理的計算機の違いを吸収するために、ミドルウェアによって実現されることが多い。
* グリッドコンピューティングに必要な要件を以下に示す。
  - 高速な計算能力
  - 巨大な記憶資源
  - 計算資源へのアクセスの容易性

2) グリッドコンピューティングの分類

* グリッドコンピューティングは、以下の種類に分類される。
  - プロセッシンググリッド：計算処理をグリッド化
  - データグリッド：データストレージをグリッド化
  - ビジネスグリッド：ビジネスシステムグリッド化

3) グリッドコンピューティング ミドルウェア

* Globus ツールキット　http://www.globus.org/
  - 事実上のグリッドコンピューティングミドルウェアのデファクトスタンダードである。オープンソースである。
  - プロトコルとしては、資源管理(GRAM、Grid Resource Management Protocol)、情報サービス(MDS - Monitoring and Discovery Service)、データ移動と管理 (GASS - Global Access to Secondary storage)、そして FTP である GridFTP を提供する。
* SCore
  「II-10-8 SCore の導入」を参照のこと。
* Sun Grid Engine
  - Sun Microsystems が主導するオープンソースのグリッドコンピューティングミドルウェア
* Xgrid
  - Apple が開発するプロプライエタリなミドルウェア。MacOSX 専用である。

4) 国家プロジェクトとしてのグリッドコンピューティング

* D-Grid (ドイツ)　http://www.d-grid.de/
* National Research Grid Initiative（日本)　http://www.naregi.org/
* GARUDA（インド)　http://www.garudaindia.in/
* The National Grid Service (英国)　http://www.grid-support.ac.uk/
* Open Science Grid(米国)　http://www.opensciencegrid.org/
* China National Grid Project(中国)　http://www.cngrid.org/web/guest/home