Reedbush 利用講義「並列数値計算」

須田礼仁

東京大学情報理工学系研究科

１．はじめに

「並列数値計算」は情報理工学系研究科の大学院講義である。隔年で実施しており，平成 28 年度にも S（夏）学期に開講している。英語名は（今回から日本語名にあわせ）“Parallel Numerical Computations”としている。講義はスライド・説明とも英語で実施しているため，留学生が多く参加している。今年度は情報理工学系研究科，工学系研究科のほか，理学系研究科,総合文化研究科の学生を含め約 60 名が履修登録した。うち 10 名は交換留学生とみられる。また，今回から学部交換留学生向けに「情報科学特別講義Ｉ」としても開講した。その結果２名が履修登録したが，うち１名は全学の交換留学生とみられる。

２．講義の概要

計算機の性能向上は並列性によって実現されている。今後まだコア数は伸びてゆくと思われるので,並列計算は誰にとっても重要な技術となってきている。計算機アーキテクチャは Intel の MIC に基づき，AVX512 を搭載する Intel Knights Landing プロセッサから構成される Oakforest-PACS，Intel Xeon を CPU，NVIDIA Pascal GPU をアクセラレータとして持つ Reedbush のように，広がりを見せている。近年はさらに FPGA の性能も向上し，解く問題によっては最新の CPU や GPU を凌駕する高い性能を達成している。

このような背景のもと，本講義では並列計算の一般論と各論とを分離し，一般論として述べられる点はできるだけ一般的な記述をするように構成した。これにより一般論が統一的な視点で理解できるようになった。ただし，一般論部分は話が抽象的になりがちで実感がわきにくくなっているかもしれず，今後とも継続的な改善を要するところである。各論においては，特に大規模並列を念頭において MPI，OpenMP，CUDA の 3 種類のプログラミングモデルを取り上げている。また，今回から CPU の SIMD の例として AVX の説明を短く入れることとした。構成としては，分散メモリの MPI を最初に説明して，OpenMP と CUDA はやや軽めに説明している。時間的には半年の講義ではこれで限界と思われるため，いくつかの重要なパラダイムが組み込めていない。ひとつは単体性能の最適化が十分に説明できていない。命令レベル並列性，キャッシュ最適化など，部分的には説明しているが，これらの HPC（ハイパフォーマンスコンピューティング）の基礎技術が十分に説明できないうらみがある。もうひとつの側面として，近年注目を浴びているタスク型の並列性，PGAS などのメモリモデル，FPGA といった新しいプロセッサについては説明できていない。同じ情報理工学系研究科において，共有メモリ並列処理でタスク型のモデルもきちんと取り上げている講義があり，多くの学生はそちらも受講してくれているようである。しかし，情報理工学系研究科の中で FPGA による高性能計算を教えている講義はないかもしれない。FPGA は数値計算で必要とする倍精度浮動小数点数は得意としていない点がネックであるが，注目されているアーキテクチャであるので，実際に触って演習することもできる講義があればよいのだが。

３．講義の内容

平成 30 年度には 13 回の講義を実施することができた。以下に各回の内容の概要を示す。

第１回：イントロダクション

この講義で取り上げる並列計算とは何かを，類似性のある並行計算，分散計算と対比する形で定義した。特に，この講義では再現性があり予測可能な計算を取り上げ，動的な並列性は最小限とする。続いて，ハードウェア並列性の基礎概念として，複数演算器とパイプライン，SIMD と MIMD，分散メモリと共有メモリ，UMA と NUMA，およびそれらのハイブリッド，均一性・不均一性などを導入した。また，この講義では所要時間を主なメトリックとすること（スループットではない），そこから高速化率，並列化効率，理想高速化率，スーパーリニアーなどの概念を導入した。また，強スケーリングと弱スケーリング，アムダール則とグスタフソン則といった相対性能の基本法則を導入した。加えて，絶対性能として flops およびピーク性能を説明した。また，所要時間測定時の注意点についても説明した。

第２回：並列性と依存性

並列性とは依存性のないこと，よって並列性の解析は依存性の解析に他ならないことからスタートした。続いて，データ依存と制御依存，RAW/WAR/WAW 依存などの依存性の類型を示した。配列，リスト，木などのデータ構造へのアクセスの並列性，ステンシル計算と超平面法・ループスキュー，木によるリダクション，カスケードによるスキャン，3 項漸化式の並列性，パイプラインなどの依存性と並列性を事例として挙げた。一方で計算順序が変えられる場合にはリオーダリングにより並列性が変わること，その表現としてカラーリングが便利であることも示した。また配列への間接アクセスがある場合を取り上げ，その並列性の抽出方法には多数ありうることを示した。

第３回：局所性

並列計算は高性能を目指すものであるが，性能という観点では局所性は避けて通れない重要事項である。まずは分散メモリでも共有メモリでも局所性が低いと高い性能が期待できないことを説明した。局所性には計算強度と粒度の２つの側面がある。まず計算強度や B/F 値について説明し，行列積を例として，プログラムの書き方次第で計算強度が変わることを示した。また行列ベクトル積のような計算強度の弱い計算もあることも注意した。リダクション，FFT，ステンシル計算の計算強度について概観し，領域分割と Owner-computes-rule が局所性の観点から利点があることを説明した。次に粒度について説明し，パイプライン計算を例に並列性と粒度のトレードオフがあることを示した。最後に単純な通信性能モデルを導入し，計算強度と粒度が性能にどのように影響を与えるかを解析し，その単純化したモデルとしてルーフラインモデルを紹介した。

第４回：スケジューリング理論

この講義では離散最適化問題に分類される古典的なスケジューリング理論についても紹介をした。HPC 分野とはやや用語が異なるため，まずその点を注意した。そのうえで，タスクとス

ケジューリングの基本概念，不均一プロセッサの類別，メイクスパン，ガント図，グラハム記法などを導入した。P||Cmax に対する LPT の最悪性能比， P|prec|Cmax に対するリストスケジューリングの最悪性能比，P|intree, pj=1|Cmax に最適解を与えるレベルスケジューリング，P∞|prec|Cmax に最適解を与えるクラスタリングスケジューリングを説明した。また，クリティカルパス，タスク挿入，タスク重複などの概念を紹介した。不均一プロセッサやオンラインスケジューリングについては既知のアルゴリズムの性能を示すにとどめた。そのかわり，Divisible Load Theory およびループスケジューリングについて紹介した。

ここまでが一般論であり，このあと各論に入る。

第５回：MPI 入門

メッセージパッシング，Local View，SPMD を説明したのち，MPI の基礎の一対一通信を説明した。Eager と Rendezvous プロトコル，ブロッキング・非ブロッキングの違い，メッセージの到着順序などについて注意をしたうえで，リダクションとステンシル計算を題材に実際の簡単な MPI 並列プログラムを説明した。

第６回：集団通信

MPI の集団通信について，基本的な集団通信と，それを実現するためのいくつかのアルゴリズムを説明した。プロセッサ数とデータサイズにより，異なるアルゴリズムが最適になりうることを確認した。また Broadcast と Reduction などの双対性，Scatter と AllGather で Broadcast が実現できるなどの集団通信の組み合わせについて解説した。

第７回：分散データ構造

まず，分散メモリ並列計算におけるデータ構造のあり方の一般論を導入した。続いて，最も単純な１次元ベクトルについて，ブロック，サイクリック，ブロックサイクリック，可変長ブロック，要素ごと指定の５種類の分散方法を示した。続いて２次元の場合には列１次元分散，行１次元分散，２次元分散の選択があり，LU 分解を例題として利害得失を考察した。またステンシル計算における袖領域を説明し，遅延隠蔽，piggy-backing を説明した。さらに，疎行列ベクトル積の事例，および動的負荷分散（この文脈では動的データ分散）の概念を紹介した。

第８回：Reedbush の使い方

情報理工学系研究科所属で，「計算科学アライアンス」の特任講師である松本正晴先生に，Reedbush の使い方について説明をしていただいた。近年，大学院生でも Linux を知らない学生が増え，そもそもコマンドラインインタフェースという概念を理解していないことも多い。（前回の講義では，この講義ではじめて Linux を知って，最初は戸惑ったが，後々とても役に立った，という学生の声を聴いた。）今回は従来よりも大幅に内容を初等的にしていただき，Linux の最重要コマンド，Reedbush へのログインやデータの送受信，コンパイルの方法，バッチとキューとは何か，どうやってタスクをキューに入れるか，など，丁寧に説明していただいた上に，PC を持ってきている学生に対しては実際にログインやテストコードのコンパイル・実行までの最小限の事項についてハンズオンをしていただいた。

第 9 回：OpenMP 入門

OpenMP の入門として，global view であること，また自動並列化ではなく正しさはプログラマの責任であることを注意した。基礎的な構文を説明したのち，OpenMP で陥りがちな誤りとして，共有・私有変数の区別，レース条件，弱い一貫性とメモリ同期の必要性を説明した。あわせて，reduction 節や atomic 節も紹介した。

第 10 回：OpenMP 高性能プログラミング

OpenMP の性能低下要因とその回避・軽減策を紹介した。排他制御のための構文である atomic，critical および lock 関数の違い，ループスケジューリングの選択肢，アフィニティに関する注意などを行った。また，キャッシュ周りの性能向上手法として，パディング，タイリング，ループ交換，ループ結合，Arrays of Structures/Structures of Arrays などの手法を紹介した。また MPI + OpenMP ハイブリッド並列化の必要性について説明した。

第 11 回：CUDA 入門

アクセラレータという概念，GPU の性能特性を導入した。続いて，CUDA の基本的な書き方の説明をした。

第 12 回：GPU の高性能計算

GPU において高い性能を達成するために重要な概念を説明した。GPU のハードウェアと CUDA の並列性階層の対応を説明し，SM ごとに走るブロックの数を決定する式を示した。またスレッドを実行する機構を説明し，並列性により遅延隠蔽が可能となることを説明した。分岐命令の削減，coalesced メモリアクセスの性能，CPU-GPU 間データ転送時間について注意を促した。

CUDA の更新は早く，講義のたびに新しい GPU と CUDA の仕様を確認している。今回は Volta アーキテクチャまでが市場に出回っていたが，Reedbush に搭載されている Pascal アーキテクチャに準拠して説明をした。

第 13 回：CPU における SIMD

今回は実施回数に余裕があったため，はじめて CPU における SIMD として，Intel AVX の説明をした。最初にイントロとして，CPU の SIMD の歴史，SIMD 関連のハードウェア，SIMD 命令の概要，GPU の SIMD との比較などを説明した。次に SIMD で性能を出すためのコードの準備（データ構造やループ構造の整理，アラインメントやエイリアスなど），続いてコンパイラオプション・指示行や OpenMP 4.5 による SIMD 指示行を説明した。プリフェッチやストリーミングストアについても少し触れた。SIMD intrinsics は存在だけを紹介した。

４．課題

講義の最初の数回で，UNIX コマンドと C/Fortran プログラミングのごく初歩的な問題をその場で解かせた。「カレントディレクトリを表示するコマンドは何か」「.o で終わるファイルを全部消すコマンドを示せ」とか，「vi か emacs で，現在カーソルがある１行を消すコマンドは何か」とか，「整数変数 n と m を宣言せよ」，「Hello, World! を表示するプログラムを書け」というレベルであるが，平均すると半分程度しかできていない。これを数回やることで，UNIX と

プログラミングができないとこの講義では困るのだということが実感してもらえたようである。以下に説明する本格的な課題の前に，UNIX コマンドや C/Fortran プログラミングの基礎を確認したり，新たに学んだり（履修の要件としているので，本来はそのようなことはないはずなのだが）した学生がいたようで，この試みは成功であった。(逆に，いかに UNIX や C/Fortran を知らない学生が多いかを確認できた。一部の教員と情報を共有し、危機感を確認した。)

この講義では全部で 5 回の課題を課した。最初の 2 つの課題は各自が使える計算機により行い，これらの課題を提出した学生には Reedbush のアカウントを配布して，残りの 3 つの課題を行わせた。

第 1 回：CPU モデルと計時方法の確認

各自のプロセッサのモデルを確認し，ピーク性能を計算させた。また，計時方法を適当に選ばせ，その精度を測定により求めさせた。これが性能測定の基本となるのだが，計時そのものに時間がかかること，また精度が有限であるなどということを想像できなかった学生もいたようである。計算機の仕組みをあまりに抽象的に理解してしまうと，このような現実に思いが至らないこともあるようである。

第 2 回：Flops への挑戦

「どんな計算でもよいので，できるだけ高い flops 値を出すプログラムを書け」という課題を出した。また，これに加えて，十分長いベクトルのコピー，内積，和の性能を測定させ，それらからメモリスループットを測らせた。

前者については，様々なプログラミング言語を用いた報告が寄せられた。最も性能が低いものは 10 Mflops 以下，最高性能は 100 Gflops を超え，最大約 15,000 倍もの性能の違いが観測された（図 1 参照)。また，第 1 回の課題で求めたピーク性能と比較させたところ、ごく少数の学生がほとんど 100% に近い性能を出した一方で、ピーク性能の 1% 未満の学生も少なくとも数名いた（図 2 参照)。性能について工夫をしないと，非常に低い性能しか出ないということを理解してもらえたと考えている。なお，図については実行性能・ピーク性能とも学生が報告したものをそのまま使っているので、正確でない場合もあると考えられる。

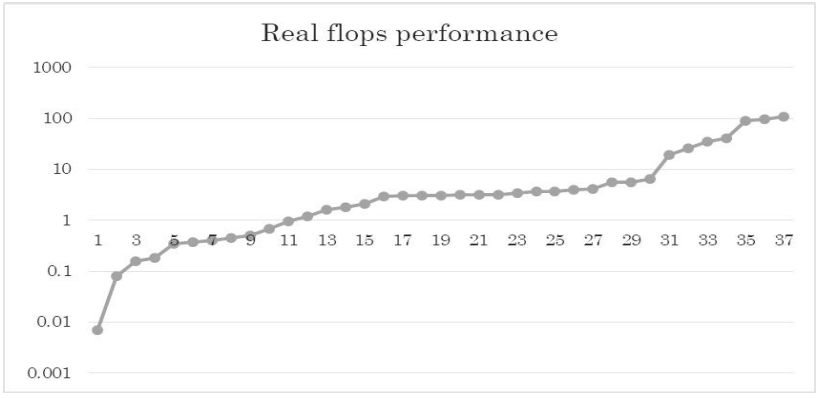

図 1：実行性能の順位プロット（X 軸：順位、Y 軸：Gflops)

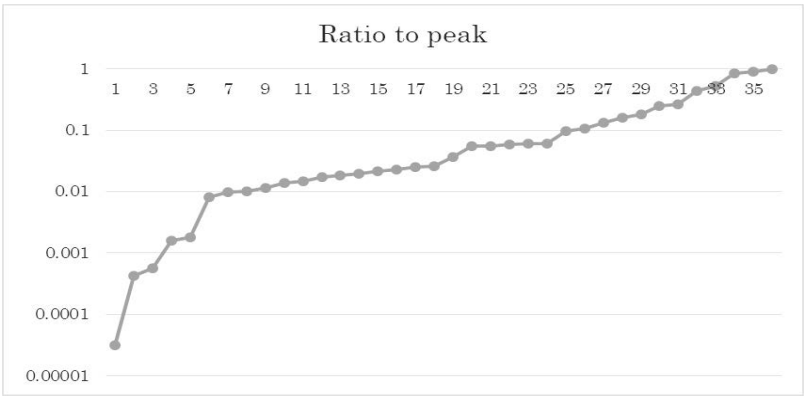

図 2：ピーク性能比の順位プロット（X 軸：順位、Y 軸：ピーク性能比）

第 3 回：Reedbush を使ってみる

Reedbush を用いて，ピンポン通信による通信性能の測定，および簡単なリダクションかステンシル計算を MPI で実行させて，Reedbush の使い方を確認してもらった。

今回は Reedbush の使い方の講義を丁寧にやっていただいたおかげもあり，大きなトラブルは発生しなかった。（自分が登録したパスワードを忘れたという学生は複数出たが。）

第 4 回：MPI プログラミング

密行列計算，偏微分方程式の数値解法，もしくは個人的に興味のある計算のいずれかを選んで，MPI でプログラミングをして Reedbush で実行し，弱スケーリングと強スケーリングでの性能を報告させた。

前回の講義ではこの課題で非常に苦戦した学生が多かったので,Cholesky 分解と熱伝導方程式については逐次プログラムをダウンロードできるようにした。まずはきちんと動く逐次コードが必要であるが，学生の専門分野によっては数値計算のコードを書いたことがないことも当然考えられ，この部分がかなりのハードルになっていたようである。その点では今回はかなりスムーズに学生たちは並列化に取り組めたようである。

講義中で注意したにも関わらず，一部の学生は計算結果が逐次計算と同じになることを確認せず，異常に高い並列化効率などを平気で報告してきた。結果を確認しないというのは、プログラミングの経験が足りないということと思われるが，それが実情なので，もう少ししつこく説明して，各自確認させるよう，今後工夫をしてゆく必要がある。「性能は測ってみないとわからない」とは言ったが，常識的に考えておかしい性能の数値は疑うことも知ってもらわなければならない。

第 5 回：OpenMP/CUDA プログラミング

今回は CPU の SIMD 機構も説明し,また計算機として Reedbush-H も使うことができたので,最後の課題は 3 つの選択肢を提示した。問題 A は主に OpenMP を用いて CPU の SIMD 機構を利用し，第 2 回の課題で行った性能試験に再チャレンジしてもらうもの。問題 B は課題 4 の MPI 実装に OpenMP を加えてハイブリッド並列にするもの。問題 C は CUDA プログラミングで

ある。第 4 回からの継続になるので、問題 B を選択した学生が多かったようである。

５．おわりに

全体として講義はスムーズに進み，Reedbush のアカウント発行も問題なくできた。今年度は前述のように，Reedbush がまったく使えない学生はほとんど出ず，Cholesky と熱伝導の逐次コードも準備したことで，学生たちは平成 28 年度の実施時よりもはるかにスムーズに課題に取り組むことができたようである。

最後になりましたが，アカウントを発行する作業をしてくださった方々をはじめ，スーパーコンピュータを用いた講義という貴重な機会を提供していただいた情報基盤センターの皆様方に感謝を申し上げます。