一週間でなれる！スパコンプログラマ

PDF 版

はじめに

世の中にはスーパーコンピューター、略してスパコンというものがある。こういうすごそうな名前があるものの例にもれず、「スパコンとはなにか」という定義は曖昧である。人によって「何がスパコンか」の定義は違うと思うが、とりあえずここでは CPU とメモリを積んだ「ノード」がたくさんあり、それらが高速なネットワークでつながっていて、大きなファイルシステムを持っているもの、と思っておけば良いと思う。

スパコンは名前に「スーパー」とついているだけに、「なんかすごそうなもの」「使うのが難しいもの」という印象を持つ人もいるだろう。しかし、スパコンを使うのに要求される技術そのものは非常に単純かつ簡単である。自分の経験として、プログラミングの素養がある学生であれば、詳しい人に一ヶ月もレクチャーを受ければ普通にスパコンにジョブを投げ始める。そのくらいスパコンを使うのは簡単なことである。しかし、スパコンは、「使いはじめる」のは簡単であるものの、「使い倒す」のはかなり難しい。経験的に、並列数が十進法で一桁増えるごとに、本質的に異なる難しさに直面する。例えば百並列で走ったコードが千並列で走らなかったり、千並列で走ったコードが一万並列でコケたりする。そのあたりの奥の深さは面白いものの、本稿では扱わない。

この記事では、近くにスパコンに詳しい人がいない人のために、「とりあえず 7 日間でスパコンを使えるようになる」ことを目指す。より正確には、「7 日間程度かければ、誰でもスパコンプログラマになれそうだな」と思ってもらうことを目指す。

Day 0： なぜスパコンを使うのか

そもそも、なぜスパコンを使うのか？それは、そこにスパコンがあるからだ。この日本語で書かれた文章を読んでいるということは、あなたは高確率で日本人であろう。おめでとう。あなたは世界有数のスパコンを使うことができる。なぜなら日本はスパコン大国だからだ。 Top500 のサイトに行くと、世界の性能トップ 500 に入るスパコンの、国の内訳を見ることができる。 2018 年 6 月時点で、トップは中国の 206 サイト、二位がアメリカの 124 サイト、日本は 36 サイトで三位に入っている。最近の中国の躍進は目覚ましいのだが、そこはさておく。 Top500 に入るスパコン数は世界三位で、しかも何度も世界一位となるスパコンを保持している日本は、世界有数のスパコン大国と言ってよいだろう。

個人的な経験で、こんなことがあった。とある海外の方と共同研究をしていた時、共同研究者に「こんな計算をしてみたらどうだろう」と提案してみた。すると彼に「そういうことができたら面白いとは思うけど、計算が重すぎて無理だよ」と言われたので「いや、うちのスパコンでやったら一日仕事だ」と言ったらえらく驚いていた。これは、単に「日本は計算資源に恵まれている」という話にとどまらず、 **人の想像力の上限は、普段使っている計算資源の規模で決まる**ということを示唆する。これは極めて重要だと思う。

普通、スパコンを使うのは、「まずローカルな PC で計算をして、それで計算が苦しくなってから、次のステップアップとしてスパコンを検討する」といった順番となるだろう。その時も、まず「これくらいの規模のスパコンを使ったら、これくらいの計算ができる」という事前の検討をしてからスパコンの利用申請をするであろう。つまり「テーマ (目的) が先、スパコン (手段) が後」となる。それは全くもって正しいのであるが、私個人の意見としては、「やることが決まってなくても、どのくらいの計算が必要かわからなくても、まずスパコンを使ってしまう」方がいろいろ良いと思う。普段からローカル PC でしか計算していない人は、なにか研究テーマを検討する際に、スパコンが必要となるテーマを無意識に却下してしまう。逆に、普段からスパコンを使いなれていると、想像力の上限が引き上げられ、普通の PC では計算できな

いような選択肢を検討できる。つまり「スパコン (手段) が先、テーマ (目的) は後」である。そもそもスパコンを使うのはさほど難しくないのだし、いろいろ検討する前に、ささっと使い始めてみよう。

注意

並列プログラミングに限らないことだが、なにかを始めようとすると、ちょっと先にそれを始めていた人がなんやかんや言ってくることだろう。「並列化効率ガー」「そもそも実行効率が悪いコードを並列化するなんて云々」とか「最初から通信の最適化を考慮云々」とか、そういうことを言ってくる人が必ず湧くが、とりあえず二年くらいは無視してかまわない。なにはともあれスパコンを使えるようになること、チューニング不足の遅いコードであろうが並列化効率が悪かろうが、とりあえずそれなりのノード数で走るコードを書いて実行してみること、まずはそこを目指そう。それなりのノード数で走るコードが書ける、それだけで十分強力な武器になる。

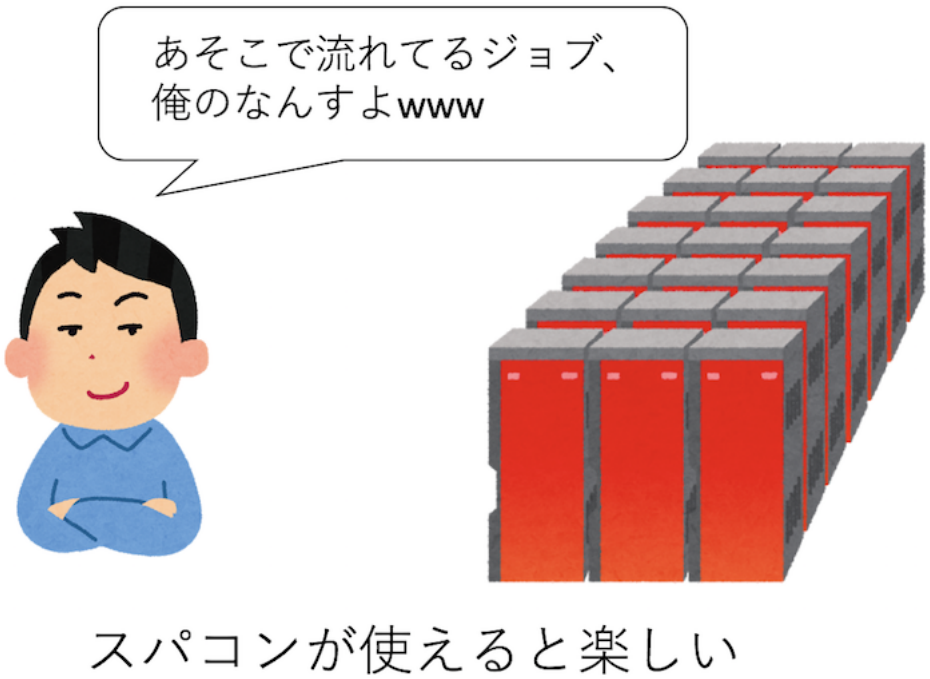

図 1: day1/fig/myjob.png

Day 1 : 環境構築［pdf］

とりえあず手元の PC で MPI が使える環境を整え、簡単な MPI プログラミングを試してみる。

- MPI とは
- 余談 : MPI は難しいか
- MPI のインストール
- はじめての MPI
- ランク
- 標準出力について
- GDB による MPI プログラムのデバッグ

Day 2： スパコンの使い方［pdf］

スパコンを使うときに知っておきたいこと。ジョブの投げ方など。

- はじめに
- スパコンとは
- 余談：BlueGene/L のメモリエラー
- スパコンのアカウントの取得方法
- ジョブの実行の仕組み
- ジョブスクリプトの書き方
- フェアシェア
- バックフィル
- チェーンジョブ
- ステージング
- 並列ファイルシステム

Day 3： 自明並列［pdf］

自明並列、通称「馬鹿パラ」のやり方について。

- 自明並列、またの名を馬鹿パラとは
- 自明並列の例 1: 円周率
- 自明並列テンプレート
- 自明並列の例 2: 多数のファイル処理
- 自明並列の例 3: 統計処理
- 並列化効率
- サンプル並列とパラメタ並列の違い

Day 4： 領域分割による非自明並列［pdf］

非自明並列の例として、一次元熱伝導方程式方程式を領域分割してみる。

- 非自明並列
- 一次元拡散方程式 (シリアル版)
- 一次元拡散方程式 (並列版)
- 余談： Eager プロトコルと Rendezvous プロトコル

Day 5： 二次元反応拡散方程式［pdf］

本格的な MPI プログラムの例として、二次元反応拡散方程式を領域分割してみる。

- シリアル版
- 並列化ステップ 1: 通信の準備など
- 並列化ステップ 2: データの保存
- 並列化ステップ 2: のりしろの通信
- 並列化ステップ 3: 並列コードの実装
- 余談：MPI の面倒くささ

Day 6：ハイブリッド並列［pdf］

プロセス並列とスレッド並列の併用によるハイブリッド並列化について。特にスレッド並列で気をつけたいことなど。

- ハイブリッド並列とは
- 仮想メモリと TLB
- 余談：TLB ミスについて
- NUMA
- OpenMP の例
- 性能評価
- 余談：ロックの話
- ハイブリッド並列の実例

Day 7：SIMD 化［pdf］

SIMD 化について。

- はじめに
- SIMD とは
- SIMD レジスタを触ってみる
- 余談：アセンブリ言語？アセンブラ言語？
- 簡単な SIMD 化の例
- 余談：x86 における浮動小数点演算の扱い
- もう少し実戦的な SIMD 化

おわりに［pdf］

謝辞

この記事は、tanakamura さんの実践的低レベルプログラミングに影響されて書き始めたものです。 angel_p_57 さんに MPI におけるバッファリングについて教えていただきました。 fujita_d_h さんには BlueGene/L の L1 エラー訂正について議論していただきました。 n_IMRC さんには、行列積における TLB ミスの論文を教えていただきました。まだこの記事が書きかけだったときにたくさん星をつけてくださった皆様、Twitter などで感想を寄せてくださった皆様、ありがとうございます。みなさんのポジティブな反応がなければ書き続けられませんでした。

本稿を読んでスパコンを使ってみよう、と思う人が一人でも増えたなら幸いです。

ライセンス