高速データ解析ライブラリ Sakura の開発

○中里 剛, 杉本 香菜子, 川崎 渉, 川上 申之介, 中村 光志, 小杉 城治
（国立天文台　チリ観測所）

概要

本稿では、我々が開発している高速データ解析ライブラリ Sakura について紹介する。Sakura は最近の CPU の特性をふまえ、ベクトル演算の徹底活用とスレッドセーフな実装により、高速な処理を実現することを目指している。現在は単一電波望遠鏡向けの機能を中心に開発を進めているが、将来的には汎用ライブラリとして公開する予定である。Sakura の性能検証を目的として、我々は Sakura をもとにした ALMA の単一鏡データ解析機能のプロトタイプを作成し、既存の機能と性能を比較した。その結果、Sakura ベースのプロトタイプは既存の機能に比べて最大で 20 倍高速であることがわかった。これらの結果に加え、プログラミングを高速化するに際して有用と思われるノウハウについても簡単に紹介する。

1. 背景

Atacama Large Millimeter/submillimeter Array (ALMA)は、日米欧の国際協力の下チリに建設され、2011 年から科学運用が行われている世界最大の電波望遠鏡である。ALMA は 62 台のアンテナからなる電波干渉計と 4 台の単一電波望遠鏡を組み合わせた大規模かつ複雑なシステムであり、1 年間にアーカイブされるデータは 200TB に達する。このような大量のデータを解析するにあたっては、大規模クラスタ計算機システムの導入など、ハードウエア面での対応とともに、解析ソフトウエアの性能を向上させることが不可欠である。

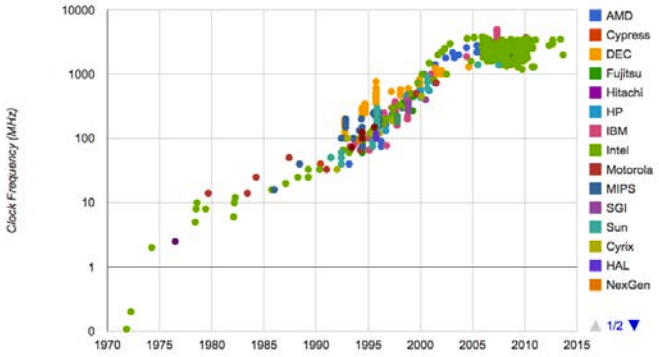

図 1 CPU クロック周波数の推移（出典：CPUDB http://cpudb.stanford.edu/）

処理を高速化することの意義は、データ解析という観点にとどまらず、たとえば望遠鏡の運用においても重要である。望遠鏡の運用においては、補償光学のようにリアルタイムのフィードバックが必要な場合がある。この場合、処理を高速化することにより、フィードバックの即時性を高めたり、あるいは近似計算に代えて（計算量の多い）正確な計算を導入することでフィードバックの精度を高めたりすることが考えられる。

ところで近年、CPU のクロック周波数は 3GHz 付近で頭打ちになっている（図 1）。最近のコンピューターの高性能化は、マルチコア化やベクトル演算機能（Single Instruction Multiple Data; SIMD）の強化など、クロック周波数以外の要素によるところが大きい。したがって、解析ソフトウエアの性能を向上させるには逐次処理では限界があり、複数のコアを同時に活用する並列処理（マルチスレッド処

理）や、ベクトル演算機能の活用といった CPU の特性を意識した開発が必要である。

2. 高速データ解析ライブラリ Sakura

Sakura は、より高速なデータ解析を目指して我々が開発を進めている高速データ解析ライブラリである。処理の高速化という観点での Sakura の特長は、(1) いくつかの特殊な機能を除き主要な機能は全てスレッドセーフな実装となっていること、(2) ベクトル演算（SIMD）を徹底活用して処理を高速化していること、の 2 点である。特にベクトル演算については、さまざまな実行環境で可能な限り高速な演算を可能とするために、Sakura を組み込んだアプリケーションの実行時に最適なベクトル演算命令を選択して処理を行う仕組みが備えられている。Sakura を利用するアプリケーションでは、スレッド並列とベクトル演算の相乗効果による処理の大幅な高速化が期待できる。現在実装されている機能を表 1 にまとめた。

表 1 Sakura の実装済み機能一覧

汎用機能	1 次元のデータ補間 最小 2 乗法によるフィッティング ビット配列の演算 しきい値に基づくデータのマスク NaN, Inf のマスク 1 次元の畳み込み（平滑化） 2 次元の畳み込み（平滑化）
専用機能 （単一鏡データ解析）	データ較正

Sakura は現在後述する ALMA 単一鏡データ解析の高速化に特化して開発されているが、将来的にはより汎用的なライブラリとして公開する予定である。

3. Sakura による ALMA 単一鏡データ解析の高速化

我々は Sakura による高速化の効果を調べるために、ALMA 単一鏡データ解析の標準ソフトウエアに Sakura を利用したプロトタイプ機能を導入し、既存機能との比較を行った。既存機能はベクトル化や並列化が有効に活用されていないが、Sakura ベースのプロトタイプ機能では、データの読み書き（I/O）をシリアルに行う I/O スレッドをひとつと、解析処理を行う複数のスレッドから構成されるマルチスレッド処理を実装した。実装の模式図を図 2 に示す。I/O スレッドはファイルからデータを読み込み、解析スレッドにデータを渡す。各解析スレッドは I/O スレッドからデータを受け取ると定められた処理フローに従って解析を行い、処理結果を I/O スレッドに返す。最後に処理結果を受け取った I/O スレッドが処理結果をディスクに書き込む。これら一連の処理が独立して実行され、全体としては I/O と複数のデータ処理が並列に進行する。プロトタイプ機能と既存機能の処理速度を比較した結果が図 3 である。図 3 に示す通り、Sakura を用いたプロトタイプ機能は、既存機能に比べて最大で約 20 倍高速な処理を達成した。

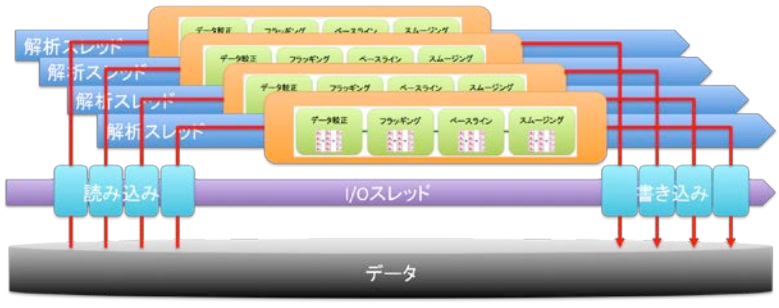

図 2 プロトタイプ機能の処理モデル

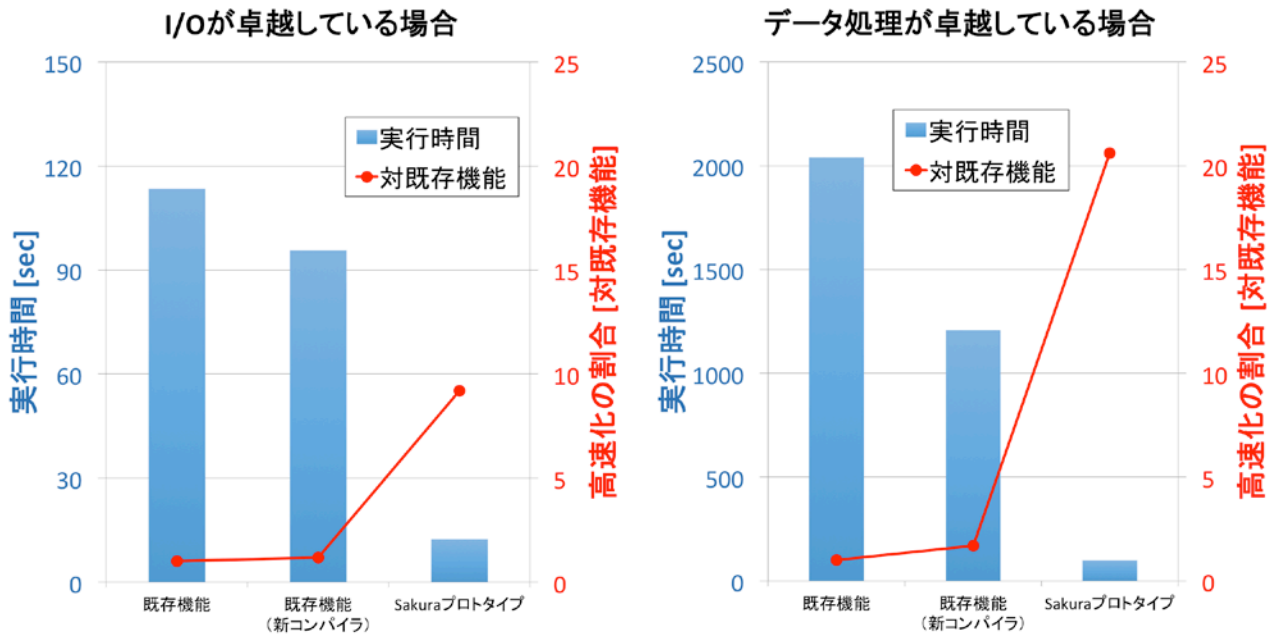

図 3 I/O の処理時間が卓越している場合（左）と、データ処理の時間が卓越している場合（右）の Sakura による高速化の効果。「既存機能」は現在使われている解析機能を使ってデータを処理した場合、「既存機能（新コンパイラ）は既存機能のソースコードを新しいコンパイラでコンパイルしたもの、そして「Sakura プロトタイプ」は Sakura ベースで既存機能と同等の機能を実装したもの。既存機能に対して、Sakura ベースのプロトタイプは左図では 9 倍、右図では 20 倍、それぞれ高速である。

4. 高速化のノウハウ

Sakura の開発および ALMA のデータ解析への応用を通じて、他のプログラムにも適用可能な高速化のノウハウを得たのでここで紹介する。

① ベクトル化のコツ

コードのベクトル化は、コンパイラの自動ベクトル化機能を利用するのが手軽でよい。その場合、コンパイラがベクトル化を行いやすいようなコードを書く必要がある。具体的には、(1) 配列の先頭アドレスを CPU が要求するバイト境界にアラインすること、(2) for ループの中に if 文を書かないこと、(3) 配列データに連続的にアクセスする（配列データに飛び飛びにアクセスしない）こと、などが挙げられる。また、コンパイラの機能を使ってプログラムが意図した通りにベクトル化されているかどうかを確認することも重要である。

② 並列化

並列化を行うに際しては、まずどの部分を並列化するかを検討する必要がある。プログラム全体を並列化できればよいが、そのようなプログラムを作るには慎重にデザインを検討する必要がある。既存のプログラムを並列化する場合、実行時間でみて大きな割合を占める部分を並列化することが重要である。たとえば、図 3 ではディスク I/O が卓越している場合（左図）は処理が卓越している場合（右図）に比べて高速化の度合いが小さくなっているが、これは並列化していないディスク I/O 部分が全体の実行時間に占める割合が大きいことによる。大して時間のかからない処理を並列化により高速化しても、全体として得られる効果は小さくなってしまう。

③　ディスク I/O の低減

一般にディスクへのアクセスはメモリアクセスやデータ処理に比べて時間のかかる処理である。データをなるべくメモリ上に保持してディスクへの書込みやディスクからの読み込みを減らすことが高速化につながる。また、ログの出力を最小限に止めることも高速化に寄与する。ただし、これらの変更はプログラムの柔軟性や利便性とのトレードオフの関係にあることには注意が必要である。たとえばログの出力を減らした場合、現在のプログラムの状態がユーザーに伝わりにくくなることにつながる。

④　新しいコンパイラを使う

コンパイラはバージョンが新しいほど賢いので、一般に同じプログラムを新しいコンパイラでコンパイルするとそれだけで高速化される。ALMA 単一鏡データ解析の場合の例を図 3 に示した。図 3 では、既存機能のソースコードを新しいコンパイラでコンパイルした後、まったく同じ処理を行って処理時間を測定した。図 3 より、単にコンパイラを新しくするだけで処理時間は 15〜30%程度減少していることがわかる。必要な作業量に比して結果として得られる効果が比較的大きいことは注目に値する。ただし、コーディング基準の違いなどによりコンパイルに失敗する場合があることには注意が必要である。その場合、コードを多少修正する必要が生じる。

5. まとめ

近年のコンピューターの高性能化を踏まえ、我々は高速データ解析ライブラリ Sakura を開発した。Sakura はスレッドセーフな実装とすることでアプリケーションレベルでの効率的な並列処理をサポートし、またベクトル演算を徹底活用して処理を最大限高速化する。Sakura を用いて ALMA 単一鏡データ解析の高速化を行った結果、既存の解析処理に比べて最大で約 20 倍処理を高速化することができた。このことは、Sakura を活用することにより ALMA のデータ解析効率が飛躍的に改善する可能性を示唆している。

また、本稿ではプログラムを高速化するための一般的なノウハウをまとめた。既存のプログラムを高速化したり、新規にプログラムを作成したりする際に参考にしていただきたい。