高速データ解析ライブラリ Sakura の開発と ALMA データ解析への応用

中里 剛[*1], 杉本 香菜子[*1], 川崎 渉[*1], 川上 申之介[*1], 中村 光志[*1], 小杉 城治[*1]

Development of High-Performance Data Analysis Software, Sakura, and Its Application to ALMA

Takeshi Nakazato[*1], Kanako Sugimoto[*1], Wataru Kawasaki[*1], Shinnosuke Kawakami[*1], Kohji Nakamura[*1], and George Kosugi[*1]

Abstract

We have been developing a general-purpose library for scientific data analysis software, Sakura. A noticeable feature of Sakura is its modern design optimum for recent CPU capabilities such as vector operation and multi-core. Based on Sakura, we have made a prototype application to reduce single dish radio telescope data taken with Atacama Large Millimeter/submillimeter Array (ALMA) for evaluation. We found that Sakura-based application is able to reduce data 10 or 20 times faster than existing data analysis software for ALMA. Our result indicates that Sakura is able to improve processing speed by making maximum use of CPU capability.

Keywords: ALMA, data analysis, high performance computing

概要

Sakura ライブラリは, 科学計算に必要な基本的な機能を提供することを目的として開発中の汎用ライブラリである. Sakura ライブラリの特長は, ベクトル演算とマルチスレッド化で CPU の性能を最大限活用し, これまでに無い高速な処理を目指している点である. 本稿では, Sakura ライブラリ導入の効果を示すため, Sakura ライブラリを基盤として大型電波望遠鏡 ALMA（Atacama Large Millimeter/submillimeter Array）の単一電波望遠鏡データを解析するアプリケーションを作成し, 既存の ALMA 用データ解析ソフトウエアとの性能比較を行った. その結果, Sakura ライブラリを使ったアプリケーションでは, 解析処理が 10〜20 倍高速化されることがわかった. この結果は, Sakura ライブラリの導入により CPU の潜在的性能を有効に活用されてアプリケーションが高速化されることを示している.

[*1] 国立天文台チリ観測所（Chile Observatory, National Astronomical Observatory of Japan）

1. はじめに

ALMA（Atacama Large Millimeter/submillimeter Array）は日本を含む東アジアと米欧の国際協力のもと南米チリのアタカマ砂漠に建設され，科学運用が行われている大型ミリ波サブミリ波電波望遠鏡である [*1]. ALMA では，取得されたデータはデータ解析アプリケーション CASA（Common Astronomy Software Applications）[*2] および CASA を基盤として実装された ALMA パイプラインシステムによって処理され，処理結果が生データとともに観測者に提供される．データレートは最大 64MB/s，平均 6.4MB/s であり，1 年間にアーカイブされるデータ量は 200TB に達する [1]．このような大量のデータを遅滞無く処理するためには，クラスター計算機の導入等ハードウエアによる処理能力強化に加え，データ解析アプリケーション自体の高速化が必須である．

ところで，最近の CPU はクロック周波数が頭打ちになっており [*3]，マルチコア化やベクトル演算幅の拡張等による処理性能の向上が主流になってきている．マルチコア CPU の場合，シリアルな処理を前提としたアプリケーションでは CPU 本来の性能を有効に活用することはできないため，高速な処理が鍵となるようなアプリケーションでは，設計・開発の段階から CPU の特長を意識し，複数のコアで並列に解析処理を行うマルチスレッド処理や，複数のデータを一度に演算するベクトル演算を活用した処理の高速化・効率化を図る必要がある. CASA や ALMA パイプラインシステムでは，比較的大きな処理単位のプロセス並列が実装されているが，各処理プロセスは基本的にシリアルな処理である [1]. ALMA のデータ解析においては，ベクトル演算や小さな処理単位でのスレッド並列処理の導入はいまだ途上にある．こうした背景から，我々はベクトル演算とマルチスレッド処理を有効活用して解析処理を高速化する汎用ライブラリの着想を得，ライブラリ名を Sakura として 2012 年からその開発に着手した．

本稿では，データ解析ライブラリ Sakura の概要とその ALMA への応用について述べる. 2 章では解析処理を高速化するという観点で Sakura ライブラリの特長を概説し，3 章で Sakura ライブラリを基盤として作成したデータ解析アプリケーションを用いた Sakura ライブラリの性能評価の結果について述べる．今回作成したデータ解析アプリケーションは, CASA に比べて 10 から 20 倍の処理速度を達成した．最後に, 4 章で今回の結果を踏まえた Sakura ライブラリ開発の今後の展望を述べる．

2. Sakura ライブラリ

2.1 Sakura ライブラリの概要

Sakura ライブラリは，基本的なデータ解析機能を提供することを目的として現在開中の汎用ライブラリである. Sakura ライブラリの大きな特長は，SIMD[*4] によるベクトル演算とマルチスレッド処理を活用した解析処理の高速化である．現状はマルチスレッドやベクトル演算による処理の高速化を検証するためのプロトタイプとして開発を進めており，高速化の実証と機能の充実を図った上で公開することを計画している．

[*1] http://almaobservatory.org
[*2] http://casa.nrao.edu/
[*3] CPU DB: http://cpudb.stanford.edu/
[*4] Single Instruction Multiple Data：単一命令で複数データを処理する並列化の手法

Sakura ライブラリの開発では基本的にコンパイラの最適化機能により処理をベクトル化するが，コンパイラによるベクトル化が不十分な場合は SIMD 拡張命令の組み込み関数を用いて可能な限り処理をベクトル化する．また，Sakura ライブラリの内部ではマルチスレッド処理を行わないが，提供する機能を可能な限り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マルチスレッド処理をサポートする．これらの工夫により，マルチコア CPU の性能を最大限に引き出すことができる．

Sakura ライブラリの概要を表 1 にまとめた．現在のところ，Sakura ライブラリは Linuxのみをサポートしている．また，コンパイラはgcc およびLLVM clangに対応し，ソースコードは C++11（ISO/IEC 14882:2011）準拠のため，gcc では 4.8.1 以降，clang では 3.3 以降が必須である（ただし clang については 3.5 でのみ動作確認済み）．他の OS，コンパイラのサポートは順次進める予定である．さらに，次節で述べるように Sakura ライブラリはアーキテクチャ毎に SIMD 命令の最適化を行っており，新しいアーキテクチャが出回ればその都度対応する．Sakura ライブラリは C 言語互換のインターフェースを持ち，ヘッダファイルをインクルードしてライブラリをリンクすればC/C++のアプリケーションで汎用的に利用できる．また，C ライブラリと連携可能な他のプログラミング言語でも利用可能である．さらに Python インターフェースを定義することにより，Python ベースのスクリプトでも手軽に利用できるようになっている．

表 1 Sakura ライブラリの概要

開発言語	C++ (C++11 準拠)
インターフェース	C, Python
依存ライブラリ	FFTW, Eigen
OS	Linux
コンパイラ	gcc 4.8.1 以降, clang 3.5 以降 (binutils 2.22 以降)
SIMD 命令の最適化	Intel SandyBridge および Haswell アーキテクチャ
規模（2014/6/19 時点）	• ソースコード行数約 17000 （うちコメント行約 2000） • ライブラリサイズ 3.3MB
計算内容 （汎用）	• 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）
計算内容 （単一電波望遠鏡専用）	• データの較正処理
その他の特長	• スレッドセーフである • メモリ管理をアプリケーション側で指定できる

2.2 SIMD 命令の最適化

SIMD は大量のデータに同じ処理を施すときに大きな性能向上が期待できるため，画像データや音声データの処理との相性が良く，マルチメディア処理では SIMD による高速化が一般的に行われている．また科学計算の分野でも SIMD の活用が進みつつある．SIMD によるベクトル化ではアーキテクチャにより利用できる拡張命令セットが異なり，一般に新しいアーキテクチャではより効率的なベクトル演算が可能である．Sakura ライブラリでは，利用できる SIMD 拡張命令セットが実行環境によって異なることを前提に，実行環境に応じた最適化を図っている．具体的には，様々な実行環境に最適化された複数のオブジェクトコードをライブラリが保持しており，実行時に最適なコードを選択することでこれを実現している．現在の実装では，SSE[*5] による最適化を行った場合と Intel AVX[*6]（SandyBridge アーキテクチャ）および Intel AVX2[*7]（Haswell アーキテクチャ）の最適化を行った場合の 3 つのオブジェクトコードがライブラリに含まれており，アプリケーション実行時に，実行環境に応じてどちらかのコードが選択される．なお，単一のソースコードに対して複数のオブジェクトコードを持つため，ソースコードの行数に対してライブラリサイズは大きくなる傾向にある（表 1）．

3. Sakura ライブラリの性能評価

3.1 性能評価の概要

Sakura ライブラリの性能評価は，Sakura ライブラリを基盤とした性能評価用データ解析アプリケーションを作成し，その性能を既存のアプリケーションと比較する，という方法で行った．比較対象のアプリケーションは CASA である．Sakura ライブラリと比較すると，CASA は 1）ベクトル化が活用されていないこと，および 2）並列化の処理単位が大きく，スレッド並列ではなくプロセス並列であることの 2 点で大きく異なっている．性能評価では，ポジションスイッチ法 [2]により取得された単一電波望遠鏡のスペクトル観測データの標準的な解析処理を性能評価用アプリケーションで実装し，CASA を用いた同等の解析処理とスループットを比較した．解析処理の概要を表 2 に示す．

表 2 解析処理の概要

処理ステップ	処理の概要
1	天体起源の電波成分の抽出 強度スケーリング
2	NaN, Inf のマスク
3	スペクトル線成分の抽出（連続波成分の除去）
4	スプリアス成分の除去
5	スペクトル線成分のスムージング
6	各データの統計量計算

[*5] Streaming SIMD Extensions: インテルが開発した SIMD 拡張命令セット
[*6] Intel Advanced Vector eXtensions: SSE 後継の SIMD 拡張命令セット
[*7] Intel AVX の機能をさらに強化した SIMD 拡張命令セット

ALMA は基本的には電波干渉計であり，その弱点を補う目的で単一電波望遠鏡が導入されている．したがってデータの解析処理は干渉計データ用と単一望遠鏡データ用との 2 つに大別され，干渉計データ解析の重みが大きい．しかし，ここでは，1)単一望遠鏡データの解析は我々日本のグループが開発を担当しており，内容の理解も進んでいて実装が容易であること，2) CASA は野辺山 45m 電波望遠鏡や ASTE 望遠鏡といった他の単一電波望遠鏡のデータ解析もサポートしており，単一望遠鏡解析の高速化は ALMA のみならず他の望遠鏡にも恩恵があること，の 2 つの理由から単一電波望遠鏡データの解析処理をサンプルとして選んでいる．干渉計データの解析については，複素数データを扱う都合上単一望遠鏡データの解析に比べると Sakura ライブラリの組み込みはそれほど単純ではないため，干渉計データ解析への応用は今後の課題である．

性能評価用アプリケーションは，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ている．実装の模式図を図 1 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受け渡す．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．

性能評価用のデータは，ALMA の評価活動で取得された単一電波望遠鏡のスペクトル観測データをもとにして作成した．スペクトルデータの総数は約 40000，各スペクトルデータのチャネル数（配列サイズ）は 3840，データサイズはおよそ 1GB である．

図 1 性能評価用アプリケーションの模式図

CASA および性能評価用アプリケーションのスループットの指標として，ここでは，単位時間あたりに処理したスペクトルデータの数を用いた．これを処理速度と呼ぶことにすると，処理速度vはスペクトルデータ数 N と処理時間 t を用いて，(1)式で表される．

$$v\ [\mathrm{sec}^{-1}] \equiv \frac{N}{t\ [\mathrm{sec}]} \tag{1}$$

性能評価に用いたデータ解析サーバの性能諸元は表 3 にまとめた．なお，この測定環境では Sakura は Intel AVX に最適化されたコードを実行する．

表 3 性能評価用データ解析サーバ性能諸元

CPU	Intel® Xeon® CPU E5-1650 @ 3.20GHz 6 cores × 1 (12 Threads)
メモリ	64GB
ディスク	3TB (512GB SSD × 6, RAID0)
OS	Red Hat Enterprise Linux Workstation release 6.3 (x86_64)
コンパイラ	gcc 4.8.1

3.2 性能評価結果

(1)式で定義した処理速度の測定結果は表 4 のようになった．また，表 4 の結果を図示したものが図 2 である．性能評価用アプリケーションではマルチスレッド処理が可能なため，アプリケーションが利用可能な最大スレッド数を 1 から CPU の最大スレッド数+1 にあたる 13（表 3）まで変えて測定を行った．また，シリアルな処理の場合についても測定を行った．一方，今回の解析処理フローでは CASA は並列化されないので, CASA の測定はマルチスレッドを無効化したケースのみである．図 2 においては，マルチスレッドを無効化した場合とマルチスレッドだが実質 1 スレッドの場合をともに最大スレッド数 1 としてプロットした．これら 2 つの測定結果を比べると，マルチスレッドを無効化した場合のほうが高速であるが，これはマルチスレッドに必要な処理がオーバーヘッドとして効いてくるためである．

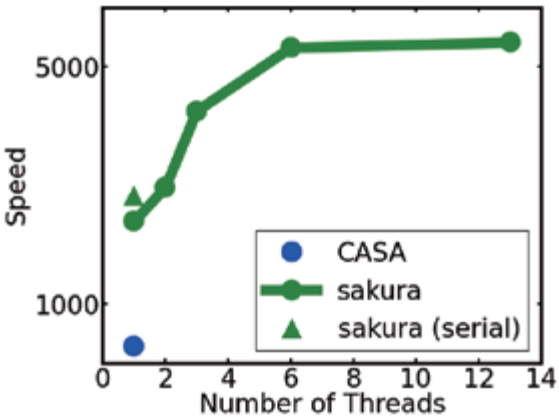

図 2 処理速度測定結果

表 4 処理速度測定結果

アプリケーション	マルチスレッド *8	処理速度v [sec^{-1}]	対 CASA 比率
Sakura	OFF	2826.74	9.79
	ON (1)	2395.24	8.29
	ON (2)	2970.82	10.3
	ON (3)	4258.72	14.8
	ON (6)	5326.35	18.4
	ON (13)	5415.30	18.8
CASA	OFF	288.71	1.0

*8 括弧内の数字は最大スレッド数を示す．また OFF はシリアル処理を表す．

表 4 および図 2 より，スレッド数により Sakura ベースのアプリケーションは CASA に比べておよそ 10 から 20 倍高速化されることがわかった．

3.3 考察

図 2 に示す通り，処理速度はスレッド数に比例して向上しない．これは処理が I/O ネックになっているためである．これを検証するため，アプリケーションの動作を変更し，処理開始時に一括でデータを読み込み，それらを解析スレッドで順次処理して処理完了後に処理結果をまとめてディスクに書き込むようにして I/O 処理と解析処理を分離した場合についても測定を行った．図 3 は I/O 処理の分離しない場合と分離した場合のアプリケーション実行時の CPU の使用率を比較したものである．図 3 において，CPU 稼働率が低い部分は主に I/O を行っている．I/O と処理が並列に行われる場合（上段）に比べて I/O と処理を分離した場合（下段）では CPU の稼働が遅れているが，これはデータの読み込みが完了するまで処理を待っているためである．図 3 より，I/O と解析処理を分離した場合，CPU の使用率は 1200%に近い（12 スレッド全てが使用率 100%に近い状態である）のに対し，分離しない場合の CPU 使用率は 300%から 1000%程度までの値を変動している，すなわち，実効的に利用されているスレッドの数が 3 から 10 程度であることがわかる．これは，I/O スレッドの処理が各解析スレッドの処理に追いついておらず，いくつかのスレッドが使われないまま待機していることを示している．なお，I/O と解析処理を分離すると，I/O と解析処理が並列に実行されないため，結果としてスループットが低下することには注意が必要である．

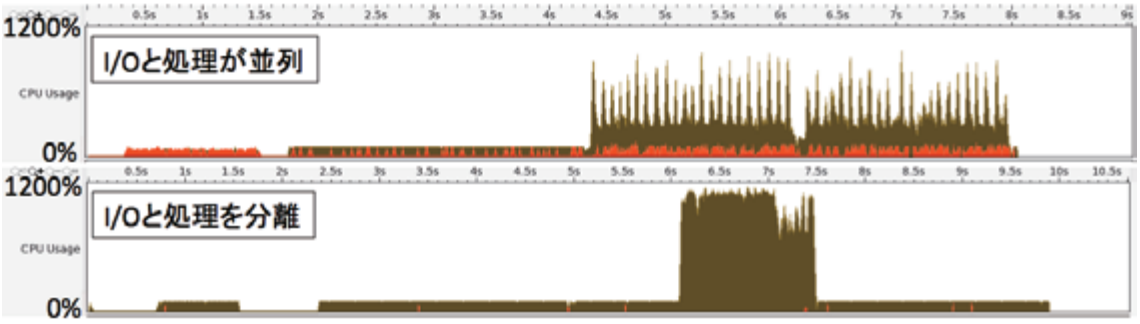

図 3 CPU 利用率の推移（最大スレッド数 13 の場合）

ところで，図 2 によれば，シリアル処理の段階でも性能評価用アプリケーションが CASA に比べて 10 倍程度高速である．これは，ベクトル化の効果[*9]に加え，コンパイラの最適化オプションの有効活用や処理内容の効率化の他，パイプライン処理により余計なデータのコピーを低減していることによる効果もある．ここで，パイプライン処理とは，各処理ステップがデータを更新しながら次のステップへ順次データを受け渡していく方式である．一方 CASA では，ユーザーの自由度を重視して各処理ステップで一旦処理を完結させるという措置を取っており，その影響で処理ステップの開始時と終了時に必ずデータの入出力処理が必要になる．今回比較対象として作成した CASA の

[*9] AVX ではベクトル演算幅が 256 ビットなので，単精度浮動小数点数（32 ビット）の演算ではベクトル化の効果は最大でも 8 倍である．

スクリプトでは，これによって生じるディスク I/O の影響を排除するため，入出力処理の代わりに各処理ステップの終わりにメモリ上に生成されたデータオブジェクトに結果を書込み，次の処理ステップで改めてデータオブジェクトからデータを読み出す形で疑似パイプライン処理を実装した．これによりディスク I/O の影響は排除できるが，それでもメモリ上でのデータの書込み・読み出しは発生しており，これによりおよそ30%程度の性能劣化が起こっている．仮にメモリ上のデータコピーをディスク I/O に置き換えた場合，性能劣化はさらに大きくなると考えられる．今後データ量が増加して I/O にかかる時間が深刻になる場合には，ある程度ユーザーの自由度を犠牲にしてでも効率化を優先してパイプライン処理を導入することも検討すべきであろう．

4. 結論

本稿では，我々が現在開発している汎用高速データ解析ライブラリ Sakura の概要と，その性能評価結果について述べた．Sakura ライブラリはベクトル演算（SIMD）とマルチスレッド処理を徹底的に活用することで CPU の性能を最大限に引き出し，処理を可能な限り高速化することを目的に開発が進められている．Sakura は C 言語互換のインターフェースを持ち，様々なアプリケーションに組み込み可能である．また Python インターフェースも定義されていて，Python ベースのスクリプトでも利用可能である．

ここでは，Sakura ライブラリの性能を評価するために性能評価用アプリケーションを作成し，ALMA のデータ解析アプリケーションである CASA との性能比較を行った．その結果，Sakura ライブラリを用いたアプリケーションでは，CASA に比べてスループットが 10 から 20 倍に向上することがわかった．スループットはアプリケーションが利用可能な最大スレッド数が大きいほど向上する．この結果は，ベクトル演算やマルチスレッド処理を活用することにより，シリアル処理を前提としたアプリケーションが大幅に高速化されうることを示唆しており，また Sakura ライブラリの導入がそのような形での高速化にきわめて有効であることを示している．

今回の測定では処理が I/O ネックになっていることもわかった．すなわち，ここで用いた解析処理フローに従って解析処理を行う限り，これ以上処理を高速化してもスループットは I/O で制限されてしまうため無意味である．解析処理を高速化すればするほど処理時間全体に占める I/O の割合は増加するため，これは当然の帰結である．従って，大規模データを扱う必要があるアプリケーションでスループットを向上させるためには，処理の高速化に加えて I/O を高速化させることがきわめて重要である．

今後は Sakura ライブラリによる高速化の実証事例として CASA への Sakura ライブラリの組み込みを本格化させつつ，Sakura ライブラリの公開に向けて機能充実，OS やコンパイラのサポート拡充を進める．これによりデータ解析や数値シミュレーションなど，様々なアプリケーションで Sakura ライブラリが活用されることを目指したい．

参考文献

1） 小杉城治, ALMA の大規模データ処理，宇宙科学情報解析論文誌，第 1 号, 2012, p.77-81 URL: http://repository.tksc.jaxa.jp/pl/dr/AA0065236010
2） 中井直正，坪井昌人，福井康雄，シリーズ現代の天文学 16 宇宙の観測 II 電波天文学，日本評論社, 2009, p.251-270