# ALMA キューブデータ簡易解析 Web アプリケーションの開発

江口智士*

# Development of a Web Application for Quick Analysis of ALMA Cube Data

Satoshi Eguchi *

**Abstract**

Virtual Observatory (VO) is a collection of various astronomical databases of various wavelengthswhich are connected by "VO interface”. Volume of astronomical observational data has been inat-ing exponentially, and the Atacama Large Millimeter/ submillimeter Array (ALMA) is estimatedto generate a 2 TB standard calibrated data cube even for one target. It is hard to handle sucha huge data cube in a personal computer. We develop a VO system which dynamically convertsa huge data cube into a small one with low resolution so that it meets users’ scientic interests.The core component of the system is ALMAWebQL, which is a typical client-server applicationbuilt on Google Web Toolkit (GWT). We developed a new endian conversion technique named\Just-in-Time endian conversion method” to reduce le reading time since the FITS format adoptsbig endian, and the endian conversion time is comparable to the (pure) le reading time and thedata extraction time on the server side. By applying this new method in native implementationto ALMAWebQL, it becomes about 3.5 ( 2.5 by switching GWT (= pure Java) to native im-plementation and 1.4 by application of multithreaded Just-in-Time endian conversion method)faster than the original GWT implementation.

Keywords: Astronomical databases: miscellaneous -- Virtual observatory tools -- Methods: data analysis

**概要**

仮想天文台 (VO) は，様々な波長の様々な天文データベースの集合体であり，それらは「VO インターフェース」という規格で結ばれている．天文の観測データは指数関数的に増加し続けており，Atacama Large Millimeter / submillimeter Array (ALMA) に至っては 1 天体につき 2TB もの標準較正済みデータが生成されると見積もられている．このようなサイズのデータは，もはや通常のパーソナルコンピュータでは処理することができない．そこで我々は，利用者の科学的関心を満足する小さなデータキューブを巨大なデータキューブから動的に生成する VO システムを構築した．この VO システムの核となるのが ALMAWebQL であり，これは Google Web Toolkit (GWT) を用いて書かれた典型的なクライアントーサーバ型アプリケーションである．また，我々はファイルの読み込みにかかる時間を短縮するために「Just-in-Time エンディアン変換法」と呼ばれる新しいエンディアン変換法を開発した．これは，FITS ファイルがビッグエンディアンを採用しており，サーバ側のデータ処理において純粋なファイル読み込みにかかる時間とエンディアンの変換にかかる時間，データを加工する時間が拮抗するためである．この方法をネイティブコードで実装して ALMAWebQL に適用したところ，GWT(=Java) 実装と比較して約 3.5 倍（GWT のみの実装からネイティブコードへの修正で 2.5 倍，マルチスレッド化された Just-in-Time エンディアン変換の適用で 1.4 倍）高速化された．

* 国立天文台天文データセンター（Astronomy Data Center, National Astronomical Observatory of Japan）

## 1. はじめに

天文の観測データは現在，指数関数的に増加している．図 1 は，様々なサーベイデータアーカイブの較正済みデータの総容量の時系列変化と今後の見通しをプロットしたものである [1]．図中の赤線で示したように，データサイズは「1 年半で 2 倍」という猛烈な勢いで増加の一途を辿っており，2020 年代に予定されている Large Synoptic Survey Telescope (LSST) に至っては，一晩の観測で 30TB もの生データが生成されると見積もられている [2]．いっぽう CPU の性能向上はここ 10 年ほど鈍化しており [3]，これまでの「観測装置から得られたデータをパーソナルコンピュータで解析する」というスタイルがまもなく破綻することは明白である．したがって，新しい解析手法の確立が急務である．

Atacama Large Millimeter / submillimeter Array (ALMA) は，日米欧の国際協力のもと南米チリに建設中の世界最大の電波望遠鏡である [4]．ALMA が生成するデータは非常に巨大であり，完成時には年間 200TB もの生データが生成される．ここから研究者が解析に使用できる標準較正済みデータに処理したものでも，1 天体につき 2TB 以上のサイズにもなる [3]．これを現在の世界のインターネットの平均速度 2.6Mbps で各自のローカルにダウンロードしようとすると，約 75 日かかる計算になる．では，適当な媒体にコピーして空輸すれば万事巧くいくかというと，先述の通り計算機の性能向上は鈍化しており，2TB ものファイルを個人のローカルの環境で処理することはほぼ不可能である．そこで国立天文台では，ALMA を来たるべき天文ビッグデータ時代のプロトタイプと位置づけ，我々 JapaneseVirtualObservatory(JVO) グループと ALMA グループが密接に連携して，VO の技術をベースにした新しいデータ解析手法の開発を精力的に行っている．

本論文では，Web ブラウザ上で ALMA のデータを簡易解析するためのアプリケーション「ALMAWebQL」の設計と，JVO ポータルサイト [5] で運用する過程で判明した問題点とその改善方法について紹介する．

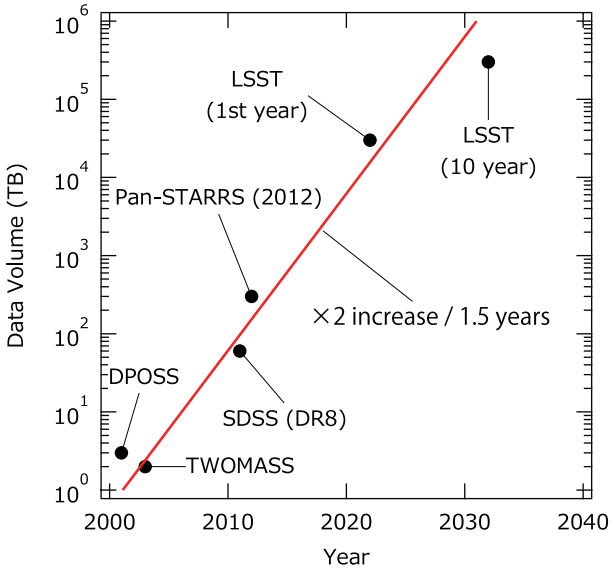


図 1 様々なサーベイの較正済みデータの総容量の時系列変化および今後の見通し

## 2. JVO ポータルサイトの設計

### 2.1 データの最適化

天文データの場合，解析する研究者の専門次第で必要とする中身が異なる．例えば，ブラックホールの観測的研究を行っている研究者は，スペクトル情報は不要だが非常に限られた領域について超高空間分解能のデータを必要とするかもし

1 1)2MASS:http://www.ipac.caltech.edu/2mass/overview/about2mass.html，2)DPOSS:http://www.ifa.hawaii.edu/~rgal/science/dposs/dposs_frames_scan.html，3)SDSS(DR8):http://www.sdss3.org/dr8/data_access.php,Pan-STARS:AlexSzalay 氏の講演資料 http://labs.yahoo.com/files/11SzalaySciAppofLargeDB.pdf より推計，4)LSST:http://www.lsst.org/lsst/science/concept_data より推計

2 http://www.lsst.org/lsst/science/development

3 http://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance

4 http://alma.mtk.nao.ac.jp/j/

5 http://jvo.nao.ac.jp/portal/

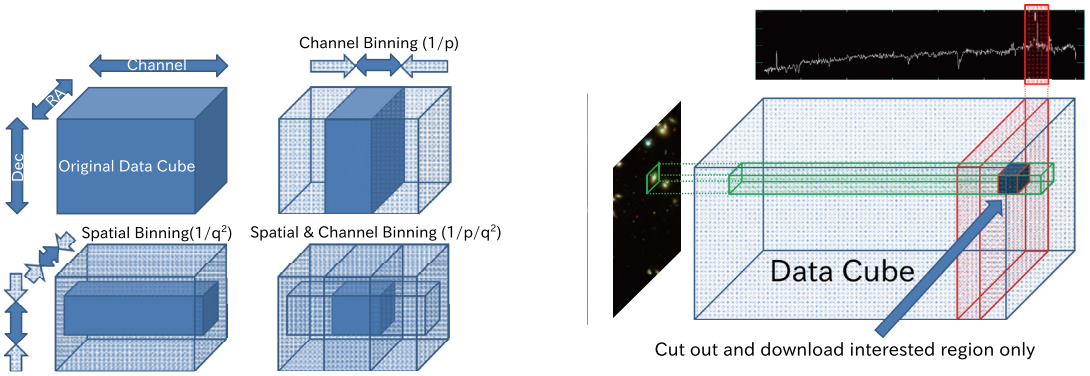


図 2　ビニングによるデータ圧縮の原理（左）と切り出し（cut-out）によるデータサイズの削減（右）

れない．いっぽう，星間物質の研究を行っている研究者の場合，ある輝線について，空間分解能はほどほどある広い領域の情報を必要とするかもしれない．このように考えると，ALMA のデータでも全部をそっくりそのまま研究に使うケースは非常に希であることがわかる．つまり，

1. ALMA の 3 次元[6] データを，様々なビニングパラメータで予めビニングしておく（図 2 左）
2. Web ブラウザでビニングパラメータを選択できるようにする．
3. 利用者の興味に応じて，選択したビニングパラメータのデータから必要な部分のみ切り出せるようにする（cut-out，図 2 右）

というシステムを構築すれば，科学的価値を損なうことなく ALMA のデータをパーソナルコンピュータで処理できるサイズまで小さくできる．

### 2.2　ポータルサイトの全体像および利用方法

以上を踏まえ，我々は JVO ポータルサイト http://jvo.nao.ac.jp/portal/ を図 3 で示される構成にすることにした[1]．ここでは ALMA のデータだけでなく，すばる望遠鏡の較正済みデータを検索・取得することができる．利用者はポータルサイトの「ALMA」のリンクから，必要なデータを観測 ID，天体名，座標等で検索する．検索結果はテーブル形式で一覧表示され，その中のうちどれかひとつのリンクをクリックすると，その観測データの詳細情報が記載されたページが表示される．このページには「WebQL」というボタンがあり，このボタンをクリックすると §2.1 の処理をグラ

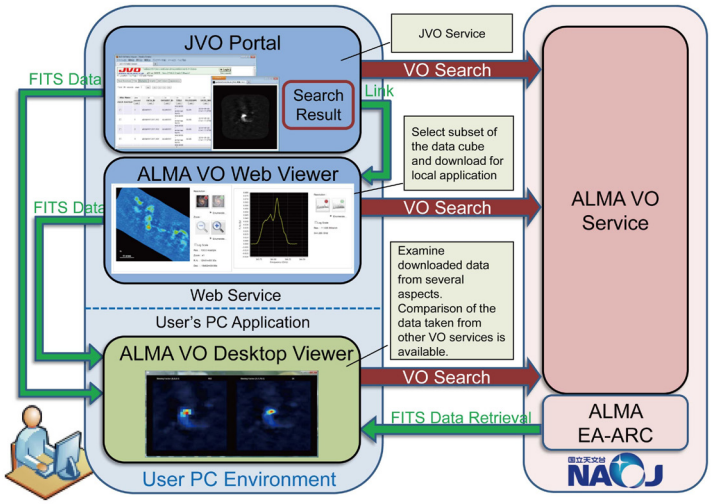


図 3　JVO ポータルサイトの全体像

6　将来的には偏光の情報を含んだ 4 次元のデータになるが，現時点で公開されているデータは 3 次元データである．

フィカルに行うための Web アプリケーション「ALMAWebQL」が起動する．利用者は ALMAWebQL を利用して，空間分解能や周波数分解能，すなわちビニングパラメータを変更したり，空間イメージを拡大・縮小したり，空間イメージの中心位置や周波数範囲を変更したりすることができる（図 4）[7]．ALMAWebQL 上で満足のいく結果が得られたら，「Download」ボタンをクリックすることで現在表示中のデータを FITS 形式で利用者のローカル環境にダウンロードすることができる．

ALMAWebQL からダウンロードしたデータは，利用者のローカル環境で動作する「Vissage」[8] を使用することにより，より詳細な解析を行うことができる．Vissage は ALMA のキューブデータを詳細に閲覧するためのソフトウェアであり，モーメント図，チャンネルマップおよび P-V 図を作成する機能を搭載している．Vissage は Java アプリケーションであるため，OS を問わず使用できる．Vissage には上記機能に加えて JVO ポータルサイトと連動する機能が実装されており，ALMA のイメージ上に JVO ポータルサイトで公開しているすばる望遠鏡のイメージをオーバーレイ表示することや，Vissage で詳細解析中にデータの不足を感じた場合に，Web ブラウザを起動してそのデータを切り出したパラメータで ALMAWebQL を直接呼び出したり，あるいは違うビニングパラメータのデータを直接ダウンロードしたりすることができる．したがって，ALMAWebQL と Vissage を行き来することにより，最終的に利用者は研究に必要十分なデータを短時間で得ることができる．

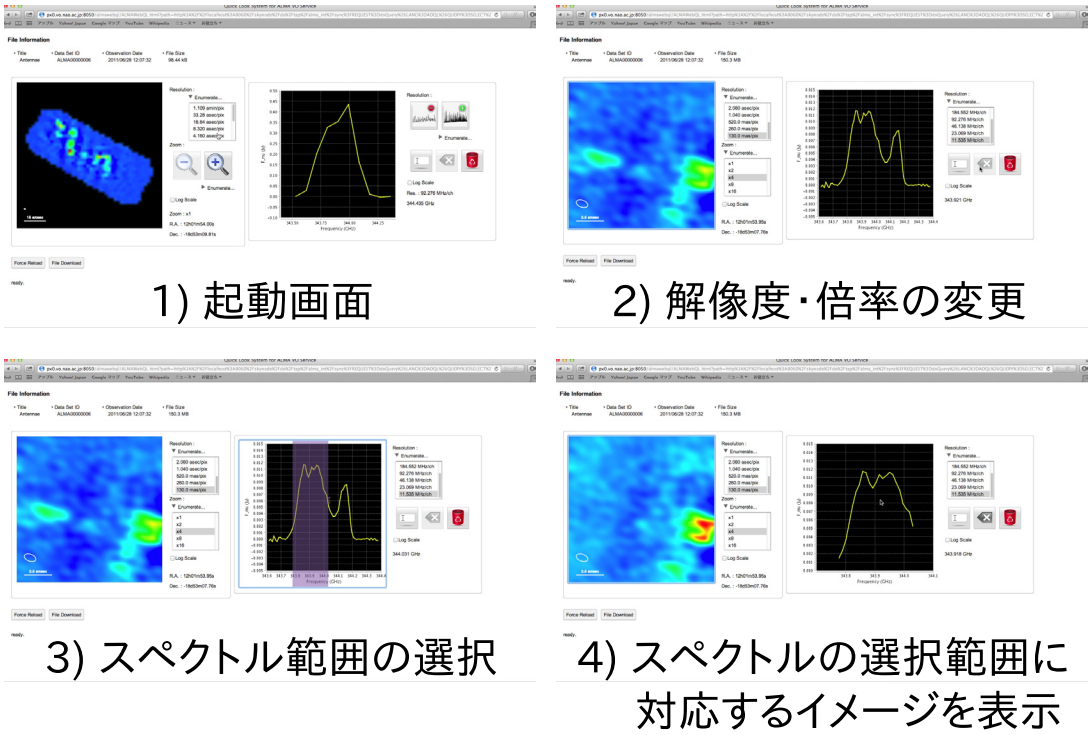


図 4 ALMAWebQL のスクリーンショット

## 3. ALMAWebQL の設計

以上のように，ALMA のデータを JVO ポータルサイトで配信する上で ALMAWebQL は極めて重要な役割を担っている．ALMAWebQL の開発目的が「ALMA の巨大データを利用者の研究目的に必要十分なサイズまで小さくする」ことであるので，すべての処理を Web ブラウザだけで行うのは本末転倒であり，ALMAWebQL の構造が JVO ポータルサイトを運営するサーバ上で動くコンポーネントと Web ブラウザ上で動くコンポーネントに分離されるのは自明である．このようなクライアントーサーバ型アプリケーションを実装する場合，両者で異なる開発言語を使用するのは保

7 現時点では，ALMAWebQL が表示できるデータは JVO ポータルサイトで公開されている ALMA のデータのみである．将来的には JVO スペース（JVO ユーザ用のオンラインディスクスペース）にアップロードされたデータも ALMAWebQL で閲覧可能になる予定である．

8 http://jvo.nao.ac.jp/download/Vissage/

守性という観点から好ましくない．というのは，クライアント側とサーバ側で全く同じ機能を実装するケースを考えたとき，片方のバグ修正をもう片方にも手作業で反映しなければならないからである．

クライアントーサーバ型アプリケーションを同じ言語で取り扱えるフレームワークの代表的なものに，Microsoft が提供している「ASP.NET」と Google が提供している「Google Web Toolkit (GWT)」の 2 つがある．前者は同社が提供する .NET Framework をベースとしており，開発言語の選択肢が複数ある．しかし，すべての処理をサーバ側で行うという欠点がある．対して後者は，クライアント側・サーバ側ともに Java 言語で記述し，コンパイル段階でコンパイラがクライアント用コードを JavaScript に変換するアプローチを取っている．ALMAWebQL では，Web ブラウザ側で表示しているイメージやスペクトルに対して物理的な値を対応させる，すなわち，マウスカーソルのある位置のピクセルが天球座標のどこに対応するのか，あるいはスペクトルのどの周波数に対応するのかをリアルタイムに計算する必要がある．仮に ASP.NET を採用した場合，ユーザがマウスを動かす度に画面上の座標と物理的値とを対応させる計算処理のために，サーバとの通信が発生する．しかし GWT を採用した場合には，画面上の座標と物理的値との対応を Java コードから生成された JavaScript コードで処理させられるため，サーバとの通信を必要最低限に抑えることができる．そのため，ALMAWebQL では GWT を採用することにした．

図 5 は，ALMAWebQL の設計を概念的に図示したものである．

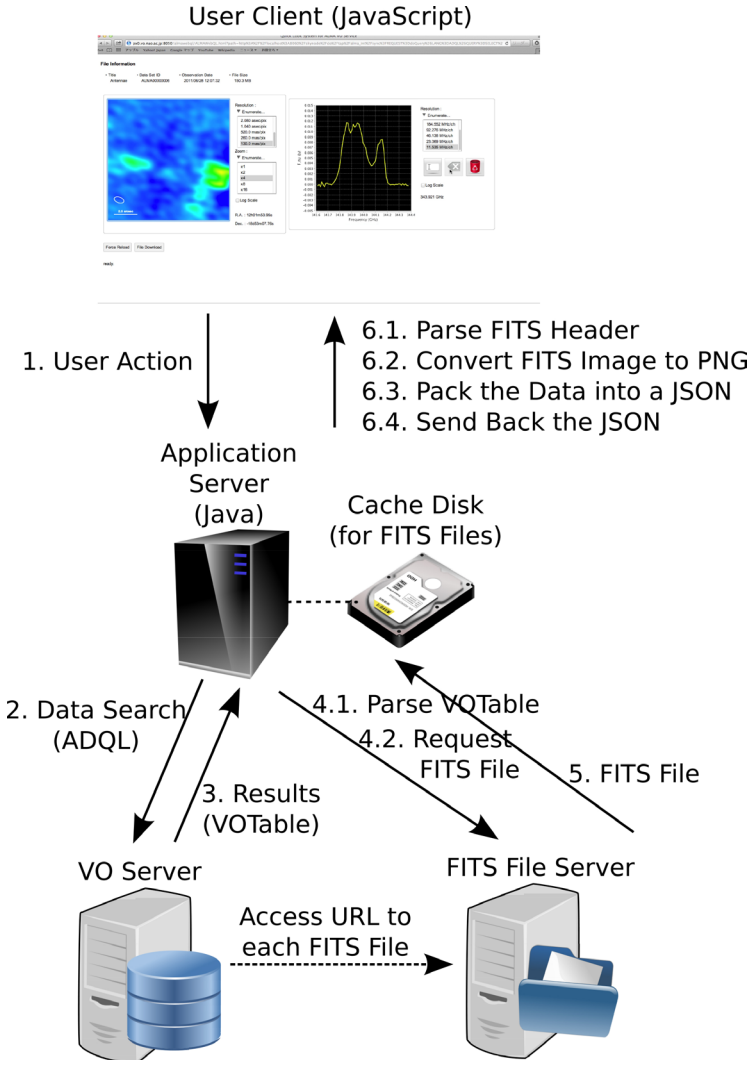


図 5　ALMAWebQL の設計概念図

1. ユーザの操作により FITS ファイルへのアクセスが必要になると，Web クライアントは Remote ProcedureCall(RPC) という仕組みを使ってサーバ側の FITS 処理メソッドを呼び出す．
2. このメソッドが呼び出されると，アプリケーションサーバは ALMA 用 VO サーバに対して ADQL と呼ばれる VO

用検索言語を用いて該当の FITS ファイルの検索を行う.

3. 検索結果は VOTable という XML 形式のテーブルで返されるので,
4. アプリケーションサーバは
   4.1. VOTable をパースして，FITS ファイルが置いてある場所（サーバ）の URL を取り出す.
   4.2. 続いてこの URL に対して HTTPGET クエリを送信して，FITS ファイルが送信されるのを待ち受ける.
5. FITS ファイルがファイルサーバから送信開始されると，アプリケーションサーバは送られてくるデータをローカルのキャッシュ用ディスクに書き込む.
6. FITS ファイルの転送が終了すると，アプリケーションサーバは
   6.1. FITS ファイルのヘッダ情報を解析すると同時に,
   6.2. イメージ HDU の中からユーザが指定した部分のみ取り出し，周波数方向あるいは空間方向にデータを積分して 2 次元イメージあるいはスペクトルに変換する[9]. できた 2 次元イメージおよびスペクトルを PNG 形式[10] で圧縮した上で Base64 でエンコードして[11],
   6.3. 両者を JavaScriptObjectNotation(JSON) 形式に変換する.
   6.4. そして，JSON 化した FITS ヘッダ，イメージおよびスペクトルを Web クライアントに送信する.
7. Web クライアントはアプリケーションサーバから送られてきたデータを元に画面を更新して, 次のユーザ操作を待つ.

**表 1 各種記憶装置の典型的な読み書き速度**

| Hardware | Read / Write Performance |
|---|---|
| Hard Disk Drive(HDD) | ~ 50 MB/s |
| Solid State Disk(SSD) | ~ 250 MB/s |
| Random Access Memory(RAM) | ~ 10 GB/s |

## 4. ファイル I/O の最適化

ALMA のデータキューブを周波数方向あるいは空間方向に積分して 2 次元イメージあるいはスペクトルを生成する処理に要する時間は，データサイズに比例する．この処理を最近の 3GHz 前後の CPU で行うと，シングルスレッドで~8GB/s の処理速度となる．表 1 は，ハードディスク，SolidStateDisk(SSD)，メインメモリの典型的な読み書き速度を表したものである．したがって，SSD を用いた RAID アレイ[12] や RAM ディスクを使用するケース，加えてオペレーティングシステム（OS）によりファイルがメモリ上にキャッシュされているケースでは，データキューブの処理速度とファイルの読み込み速度が拮抗する．特に ALMAWebQL では，FITS ファイルサーバからファイルをダウンロードしてアプリケーションサーバのローカルディスクに書き込む過程で，その内容が OS によりメモリ上にキャッシュされるため，ファイル I/O 処理を工夫することにより，ALMAWebQL のレスポンスを向上させられると考えられる．そこで我々は，エンディアン変換に注目した.

エンディアンとは，コンピュータが数値データをメモリに格納するときの順番を表す言葉で，メモリの内容を 16 進ダンプしたときに大きい位から小さい位という, 我々人間が通常使っている順番で数値が表示される方式をビッグエンディアン，その反対をリトルエンディアンと言う．現在稼働しているサーバやパーソナルコンピュータではほとんどがリトルエンディアンを採用している一方，天文の世界で広く使われている FITS 形式はビッグエンディアンを採用している．したがって，FITS データを処理する過程で必ずエンディアンの変換を行う必要が出てくる．我々はこの変換過程を最適化できないかと考えて試行錯誤した結果,「Just-in-Time エンディアン変換法」という新しいアルゴリズムを編み出した[2].

9 ALMA の FITS ファイルの構造については図 2 を参照.

10 JPEG ではなく PNG フォーマットを採用したのは，JPEG 形式は離散コサイン変換を施した後にバンドパスフィルタを通すため，いわゆる「モアレ」が発生する可能性があり，データ圧縮による偽の空間構造と本物の空間構造との区別が困難になるからである．PNG フォーマットは JPEG よりも圧縮率は悪いものの可逆圧縮アルゴリズムを採用しており，このような問題は発生しない.

11 こうすることで，アプリケーションサーバ側に外部公開用ディレクトリを作成して，そこに一時的な PNG ファイルを置くことを避けられるため，セキュリティ的に安全性が確保される.

12 市販（民生品）の SSD を 16 個で RAID0 のシステムを構成して限界までチューニングを施すと，シーケンシャルアクセスで～4GB/s の速度が出る（http://www.fumi.org/pc/ultra40/index.html）

ここで便宜上，FITS ファイルを取り扱うライブラリ，例えば CFITSIO[4] や SFITSIO[13] 等で使われている方法を「OnAhead エンディアン変換法」と呼ぶことにする．

double 型のデータのエンディアン変換を行う関数を

```
double convert_endian(double x);
```

とするとき，FITS ファイルの全要素を足し合わせるコードは次のように書ける：

•On Ahead エンディアン変換法

```
{
  double *v;        // ファイルの中身を保持する配列
  size_t len;       // 配列の要素数
  size_t i;
  double sum = 0.0;

  // ファイルの読み込み（省略）

  // エンディアン変換（注：FITS ライブラリ内部）
  for (i=0; i<len; ++i){
    v[i] = convert_endian(v[i]);
  }

  // 総和を求める
  for (i=0; i<len; ++i){
    sum += v[i];
  }
}
```

•Just-in-Time エンディアン変換法

```
{
  double *v;        // ファイルの中身を保持する配列
  size_t len;       // 配列の要素数
  size_t i;
  double sum = 0.0;

  // ファイルの読み込み（省略）

  // 総和を求める
  for (i=0; i<len; ++i){
    sum += convert_endian(v[i]);      // エンディアン変換
  }
}
```

3.4GB の 2 次元 FITS イメージを使って，上記の「総和を求める」というコメントのあるループの処理時間と，ファイルの読み込み部分も含めて全部の処理を完了するのに要した時間を測定した（図 6）．総和を求めるループのみを比較すると，Just-in-Time エンディアン変換法ではエンディアン変換を行う処理が加わっているため，On Ahead エンディアン変換法よりも有意に遅くなりそうなものであるが，実際は予想に反して両者に違いは認められない．対して全体の処理時

13　http://www.ir.isas.jaxa.jp/~cyamauch/sli/index.ni.html

間を比較すると，On Ahead エンディアン変換法のみエンディアン変換のループが存在するため，このループ分だけ Just-in-Time エンディアン変換法よりも遅くなる．結果的に, Just-in-Time エンディアン変換法を用いることで, シングルスレッドで約 20%，マルチスレッド化した場合には約 40% もパフォーマンスが向上することが判明した．なお，この比較はネイティブコードで行った．

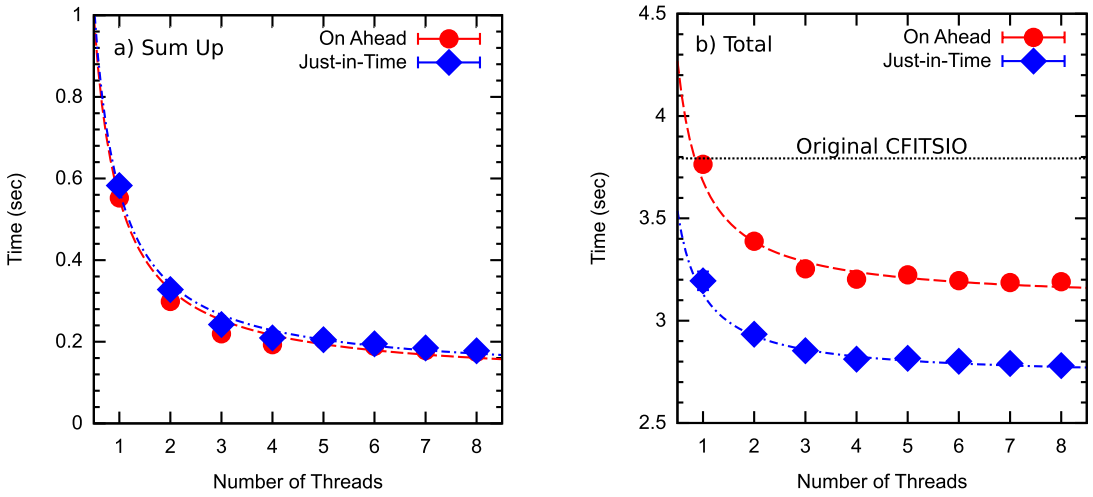


**図 6　3.4GB の 2 次元イメージ FITS ファイルの全要素を足し上げるのに要した時間のうち，総和を求めるループ部分のみを終了するのに要した時間（左）と，ファイルの読み込み時間も含めた処理時間の合計（右）**

いっぽう，Java はビッグエンディアンを採用しており，ソースコード上はエンディアン変換を行う必要はないため，仕組みを正しく理解していないとネイティブコードよりも高速に FITS ファイルを処理できそうに思えてしまう．そこで，Java と nom.tam.fits[14] ライブラリを用いて上と同じ FITS ファイルの全要素を足し上げるのに要する時間を測定したところ，ネイティブコードより約 2.5 倍遅いことがわかった．これは double 型のデータを CPU のレジスタ (Java の場合は「スタック」) に読み込むのに，ネイティブコードでは 1 オペレーションコードの実行で済むところを，Java の仮想マシンでは 4 つのオペレーションコードを実行する必要があるからである [15].

続いて，Just-in-Time エンディアン変換法を ALMAWebQL に適用して, (シングルスレッドで) ファイルサイズに対してどのように振る舞うかを調べた（図 7）[16]．イメージの生成およびスペクトルの生成ともに，100MB 付近で On Ahead エンディアン変換法よりも Just-in-Time エンディアン変換法が優勢になり，200MB 付近で予測通り約 20% 高速であることがわかる．したがって，Java から Just-in-Time 変換法を用いた並列化されたネイティブコードに切り替えることで，200MB 以上のファイルに対して

- Java からネイティブコード (OnAhead エンディアン変換法 ) に切り替えることで約 2.5 倍
- ネイティブコード (OnAhead エンディアン変換法 ) に対して並列化された Just-in-Time エンディアン変換を適用することで約 1.4 倍

の合計約 3.5 倍の高速化が達成できた計算になる．

14　NASA/GoddardSpaceFlightCenter で開発された Java 用 FITS ライブラリ

15　http://docs.oracle.com/javase/specs/

16　GWT では，アプリケーションサーバで動くコードは純粋な Java サーブレットであるので，JavaNativeInterface(JNI) を介することで，ネイティブコードで書かれた Just-in-Time エンディアン変換法のコードを使用することができる．

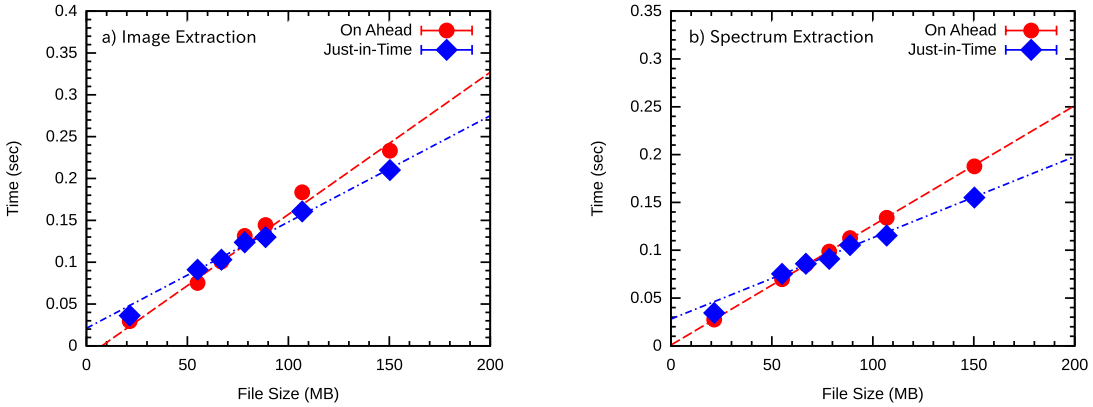


図 7　Just-in-Time エンディアン変換法を適用した ALMAWebQL の，2 次元イメージの生成時間（左）とスペクトルの生成時間（右）のファイルサイズ依存性

## 5. まとめ

現在，天文データのサイズは指数関数的に増加を続けており，現在の「すべての解析作業を個人の計算機環境で完結させる」スタイルは，非常に高い確率で近い将来破綻する．これは昨年観測が始まった ALMA 望遠鏡のデータで現実のものとなりつつあり，1 天体につき 2TB を超える ALMA のデータを解析する工夫が必要である．そこで我々は，巨大データをビニングで事前に小さくしておき，それを Web ブラウザ上で動くアプリケーション「ALMAWebQL」で簡易解析して利用者の目的に合致するデータを動的に生成する VO システムを構築した．ALMAWebQL は GWT を使って書かれた Java プログラムである．

ALMAWebQL では，データキューブを処理する時間とファイルを読み込む時間が拮抗するため，ファイル読み込みを高速化することでアプリケーションのレスポンスを向上させるアプローチを採用することにした．FITS ファイルを読み込む際にはエンディアン変換を行う必要があり，ここを新開発の「Just-in-Time エンディアン変換法」を用いたネイティブコードに置き換え，マルチスレッド化することにより，GWT だけ（=Java）で記述したときよりも約 3.5 倍ものパフォーマンス向上が得られた．

今後も ALMAWebQL の開発は精力的に行う予定であり，新しい機能ができ次第順次公開する予定である．是非一度 JVO のポータルサイト http://jvo.nao.ac.jp/portal/ を訪れて実際に ALMAWebQL を使って頂き，機能改善・要望等を送って頂けると幸いである．

## 参考文献

1) Eguchi, S., Kawasaki, W., Shirasaki, Y., Komiya, Y., Kosugi, G., Ohishi, M., Mizumoto, Y., Pro-totype Implementation of Web and Desktop Applications for ALMA Science Verication Data and the Lessons Learned, arXiv:1211.3790 (2012 November)
2) Eguchi, S., \Superluminal" FITS File Processing on Multiprocessors: Zero Time Endian Conversion Technique, Publications of Astronomical Society of the Pacic, Vol. 125, No. 927 (2013 May), pp.565-579
3) Lucas, R., Richer, J., Shepherd, D., Testi, L., Wright, M., & Wilson, C., Estimation of ALMA Data Rate, ALMA Memo #501 (2004 June)
4) Pence, W. D., CFITSIO: A FITS File Subroutine Library, Astrophysics Source Code Library, record ascl:1010.001 (2010 October)