ACA GPU分光計の開発 - ソフトウェア品質向上の取り組み

○清水上 誠、渡辺 学、臧 亮堅、中山 進、小杉 城治
(自然科学研究機構 国立天文台 アルマプロジェクト)

概要

Korea Astronomy and Space Science Institute (KASI) と国立天文台アルマプロジェクトでは、ALMA 望遠鏡アタカマコンパクトアレイで観測に供する分光計の開発を行っている。この分光計は GPGPU を用い、CUDA ライブラリと C++ 言語で実装されたソフトウェアによって、384Gbps の入力信号を、実時間で処理する。本発表では、この分光計のソフトウェア開発における、品質向上のための取り組みとそれを実現するツールについて紹介する。

1. 導入

1.1. ACAトータルパワーアレイ

ALMA 望遠鏡は、南米チリ共和国の北部、標高 5000m のアタカマ砂漠に位置する電波望遠鏡である。ALMA 望遠鏡を構成するアンテナ 66 台のうち、16 台が日本が開発したアンテナで、この 16 台によってアタカマコンパクトアレイ (ACA、別名モリタアレイ) を成す。ACA は 12 台の 7m アンテナと 4 台の 12m アンテナから構成され、4 台の 12m アンテナはトータルパワーアレイと呼ばれ、天体からやってくる電波の強度を精密に測定することを目的としている。

1.2. ACA分光計

ACA 分光計は、ACA トータルパワーアレイを構成している 4 台のアンテナからの信号を入力とし、同一アンテナにおける偏波ごとの自己相関スペクトルと偏波間の相互相関スペクトル、および、アンテナ基線毎のアンテナ間の相互相関スペクトルを、実時間で求めるシステムである。ACA 分光計は、4 ベースバンド × 4 アンテナ × 2 偏波 × 3bit (量子化ビット数) × 4GHz (サンプリングレート) = 計 384Gbps の時系列信号を入力とし、4 ベースバンド × {4 基線 × 3 偏波対 × 4-8Byte ＋ 6 基線 × 4 偏波対 × 8Byte} × 64 スペクトルウィンドウ × 8192ch × 1KHz = 最大 136Mbps でパワースペクトルの出力を得る。なお、出力のパラメーターである偏波対・スペクトルウィンドウ・チャンネル数・出力周期は最大のビットレートを超えないよう、観測設定が行われる。

1.3. システム構成

ACA 分光計システムは、実質的な処理を担う、ACA Spectrometer Control (ASC) ・ACA Spectrometer Module (ASM) と、付随する、Erbium Doped Fiber Amplifier (EDFA) ・光スプリッター・ネットワークスイッチなどを、構成要素とする。

1.4. ASM

ASM は、ACA 分光計の計算処理を担う装置である。ASM には、計算機とその計算機上で動作するソフトウェアが含まれ、必要がある場合には ASM 計算機や ASM ソフトウェアとして呼び分ける。ASM は、1 台のコールドスペアを含む 5 台あり、それぞれの ASM は、5U ラックマウント型の汎用の計算機に、光信号を受信する 2 枚の DRXP ボードと、計算処理を行う 4 枚の GPGPU ボードを、搭載している。1 台の ASM が 4 つのアンテナの 1 つのベースバンドの処理を担い、4 台に分散して、4 ベースバンド × 4 アンテナの、分光処理を行う。

2. ASMソフトウェアの開発

2.1. 開発手法

ASM ソフトウェアにおいては、モジュール構成についても計算処理についても、初期には実現方法や実現性が確定的ではない部分が多くあり、プロトタイピングによって方針を確定していく必要があった。そのため、ASM ソフトウェアの開発は、長期のスケジュールの見積りには概算工数の積み上げを用いながらも、中短期には反復増加型で行うこととした。

ここでの反復増加型開発の特徴は、全ての機能を実現した製品を 1 度にリリースせず、機能を順次追加してリリースを繰り返すことである。ASM の場合は、数日から数週間をリリースサイクルとし、各リリースは、スケジュールや技術的難易度を考慮して、分割可能な機能単位で、期限と順番を動的に決めている。

リリースという形で進捗を共有することで、プロジェクト内部での中短期のスケジュールを組み立てることが可能となっている。また、実行可能な形で各リリースを行うことで、ASM ソフトウェアのコンポーネント試験および他コンポーネントとのインテグレーション試験を、開発者以外も行うことが可能である。

3. 設計と実装

ASM ソフトウェアにおいては、反復増加型開発により設計と実装それぞれの更新頻度が高く、プロトタイピングによって実装から設計へのフィードバックも頻繁に行われる。設計と実装の品質を向上させるためには、それらを相互に遅滞なく更新し、齟齬が無い状態を保つ必要がある。これを実現するために、ソースコードツリーからの文書生成、プレーンテキストによる文書と図の記述、バージョン管理による変更の追跡を、体系的に行うことに取り組むこととした。以下に利用したツールと実現される効果を紹介する。

3.1. Doxygen

Doxygen は、ソースコードツリーから文書を生成するツールである。生成する文書は、Javadoc の様な HTML 形式や PDF 形式などを選択することができる。Doxygen は、ASM ソフトウェアの実装で用いる C++ 言語に対応しており、C++ 言語のヘッダファイルやソースファイルを解析し、ネームスペース・クラス・構造体・共用体・関数・変数・定数などの要素について、関係を

考慮して、Application Programming Interface (API) 文書を自動生成する。また、ソースコード上に記載したコメントも収集され文書に組み込まれるため、設計意図などを加えることもできる。ASM ソフトウェアでは生成される文書を詳細設計文書として利用することとした。

3.2. HTML & Markdown

HTML と Markdown はどちらも書式付き文書を記述するためのマークアップ言語である。複数のクラスから構成されるモジュールの関係や静的に解析できない実行時のシーケンスなど、抽象度が高い設計は、自動生成を利用して文書化することはできず、個別の記述が必要になる。Doxygen は、ソースコード以外に HTML や Markdown を文書に組み込むことも可能であるため、抽象度が高い設計については、HTML や Markdown で記述し、ソースコードツリーに含めて管理することで、Doxygen によって集約することした。HTML は CSS と JavaScript と組み合わせることで複雑な表現を実現することができる。

3.3. Graphviz & PlantUML

Graphviz は、DOT 言語で記述されたダイアグラムを描画するツールである。また、PlantUML は、PlantUML 専用言語で記載した Unified Modeling Language (UML) を描画するツールで、Graphviz のフロントエンドとして動作する。ASM ソフトウェアの設計においては、シーケンス図・クラス図・オブジェクト図などでの図示に PlantUML を用いることとし、PlantUML 専用言語で記述した。DOT 言語および PlantUML 専用言語は、ソースコード・Markdown などの一部として記述することができ、Doxygen によって自動的にレンダリングされ、文書に組み込まれる。また、Doxygen が解析した構造の図示にも利用され、ソースコードに紐づくクラス図やコールグラフなどの図は自動的に生成される。

3.4. Git

Git は、ソースコードなどの変更履歴を記録・追跡するための分散型バージョン管理システムである。ASM ソフトウェアの開発では、ソースコードはもとより、文書や図をプレーンテキストで記述することで、Git を使った変更の追跡とバージョン管理を可能とした。

4. 設計と試験

ASM ソフトウェアにおいては、試験を容易に行えることを、一つの設計指針のとして据えている。より具体的には、プログラミングパラダイムを背景とした、関数分離・クラス継承・副作用排除や、Linux 哲学を背景とした、単機能特化・フィルタープログラム・階層的思考などの考え方を取り入れている。これらの考え方を取り入れた設計を行うことで、単純な入力とそこから得られる出力の予測が容易になるため、試験の設計や実装も容易になる。容易な試験は自動試験につながる。ASM ソフトウェアの開発では、品質向上のため、開発者による単体試験の実装と実施、自動的な継続的インテグレーションによる早期の不具合発見に取り組むこととした。以下に利用したツールと実現される効果を紹介する。

4.1. Google C++ Testing Framework

Google C++ Testing Framework (Google Test) は C++ 言語を用いるプロジェクトで単体試験を行うフレームワークである。Google Test は、ライブラリとして提供され、Google Test のテストランナーを起動する main 関数を C++ 言語で実装し、コンパイルした後、実行ファイルを実行することで単体試験を行う。単体試験のテストケースは、Google Test が提供する Assertion マクロの中から適切な物を選択し、C++ 言語のソースコードとして記述し、プロジェクトのソースツリーに保存する。試験を実行すると、選択した Assertion マクロの種類に沿って、状態や変数が、想定される結果と比較され、試験の合否の出力が得られる。Google Test の合否の出力は、カラーコード付きの標準出力の他、JUnit 形式の xml ファイルでも取得できる。JUnit 形式の xml は結果の自動解析や試験文書の生成に用いることが可能である。

4.2. Jenkins

Jenkins は自動化サーバで、ソフトウェア開発においてビルド・試験・デプロイなどを自動的に行う継続的インテグレーションサービス (CI) を提供する。Jenkin は多くの場合はサーバーサイドで実行され、ユーザーは WEB インターフェースを介して操作する。Jenkins はリポジトリへのコミットのタイミングや定時にプロジェクトのビルドや試験などを行い、結果は WEB インターフェース・レポート文書・メールなどで確認することができる。自動的にビルドや試験が実行され結果が記録されることで、不具合が混入した場合には、結果の履歴を遡り、問題箇所を容易に特定することが可能になる。ASM ソフトウェアの場合は、Git リポジトリへのコミットをトリガーにして、eclipse によるヘッドレスビルド・Make によるビルド・単体試験を自動実行している。

5. まとめ

ASM ソフトウェアは、実行可能な形式で機能追加のリリースを頻繁に行う、反復増加型の開発手法を取っている。開発者以外のメンバーが早い段階でソフトウェアを使用し、客観的に評価や試験を行うことが可能となっており、ASM ソフトウェアの品質向上に寄与している。

また、ASM ソフトウェアの開発では、設計・実装・試験の一連の工程で、自動化のためのツールを導入することで、次のようなことを実現している。

1. ソースコードを元に文書を生成することによる、実装と矛盾の無い設計書や説明書の維持。
2. 文書や図をプレーンテキストで記述することによる、ソースコードと同様の管理。
3. バージョン管理ツールを使用することでの、ソースコード・文書・図の容易な変更の追跡。
4. 自動試験のための、容易な試験と単純な設計。
5. 継続的なビルドと試験の監視による、積極的で高頻度の設計や実装の更新。

ASM ソフトウェアの開発では、これらを実現することで、文書・図・ソースコードと最終的な製品の品質向上に取り組んでいる。