SPring-8 における新しい制御フレームワーク MADOCA Ⅱの開発

DEVELOPMENT OF NEW CONTROL FRAMEWORK MADOCA II AT SPRING-8

松本崇博#, A), 古川行人 A), 石井美保 A)
Takahiro Matsumoto #, A), Yukito Furukawa A), Miho Ishii A)
A) Japan Synchrotron Radiation Research Institute (JASRI)

Abstract

MADOCA (Message And Database Oriented Control Architecture) was originally developed at SPring-8 on 1995, and has been successfully utilized in SPring-8 and also in other facilities, HiSOR, NewSUBARU and SACLA for controls of accelerator, beam line and data acquisition system in experiments. However, we concluded several expansions in its functionalities are needed to satisfy requirements in our future upgrades in control systems. Therefore, we developed next generation MADOCA control framework, MADOCA II this time. In this proceeding, we report on MADOCA II from the aspect of the messaging. The main features of MADOCA II are as follows; 1) Variable length data such as a waveform data and an image data can be transferred with a message. 2) Asynchronous controls can be performed for fast processing of multiple controls. 3) Control system can be operated on Windows as well as Linux and Solaris. These functions were implemented by replacement of the used messaging library by ZeroMQ. We already started to introduce MADOCA II for several applications in SPring-8. The status of the replacement of our control system with MADOCA II is also reported.

1. はじめに

MADOCA は分散制御システムのための制御フレームワークであり、1995 年に SPring-8 の加速器やビームラインを制御するために SPring-8 において独自に開発された[1]。現在では、HiSOR、NewSUBARU、SACLA など他の放射光施設における制御システムにも利用されている。SACLA では加速器制御と共にビームライン、及び実験ステーションでのデータ収集(DAQ)の制御においても MADOCA が広範囲に適用されており、MADOCA はこれら大規模な施設において順調に運用されてきている。

しかしながら、今後の SPring-8 加速器の高度化や、SPring-8 II における制御系への要求も考慮した際に、いくつかの課題が見えてきた。これらを改善するために次世代の MADOCA 制御フレームワーク、MADOCA II を開発した。本稿では MADOCA II のメッセージングにおいて新しく拡張された以下の項目に重点を置いて述べる。

1. 可変長データへの対応
2. 制御用端末とフロントエンド計算機間の通信の非同期化
3. Windows による制御

近年、加速器のビーム診断等においては、波形データやカメラ画像など多様なデータを扱うことが日常になってきており、制御における可変長データへの対応が重要になってきている。また、扱う機器の多様性が増加し、Windows でのみ扱うことができるドライバーを扱う機会も増えている。現状を鑑みると、MADOCA II のメッセージングにおいて、これらの機能が実装されることの意義は大きい。
メッセージングの他にもデータ収集とデータ蓄積に関して改良を検討しているが、こちらの状況については[2]を参照されたい。

本稿では、はじめに MADOCA の概要について述べ、現在抱えるさまざまな課題について示す。その後、MADOCA II で拡張された新機能について述べる。MADOCA II は既にいくつかの事例で SPring-8 の制御系に導入されつつある。これらについて紹介すると共に、現在、SPring-8 及び SACLA において進められている MADOCA II 制御系への移行に関する状況についても触れる。

2. MADOCA が現在抱えるさまざまな課題

MADOCA における制御の概略図は Figure 1 に示される。制御命令を行う GUI(Graphical User Interface)部と、ハードウェアを制御する EM(Equipment Manager)部のレイヤーがそれぞれ分離されている。GUI が設置される制御用端末(クライアント側)と、EM が設置されるフロントエンド計算機(サーバー側、主には VME) は複数あり、ネットワーク通信を介してこれら分散化されたシステムを制御できるようになっている。

GUI 側からの機器への制御指令は人間が理解しやすい SVOC 形式のメッセージを経由して送られる。例えば V/O/C = get/sr_rf_cavity/voltage のメッセージの場合は、Verb(動作)が get、Object(制御対象の機器名)が sr_rf_cavity 、Complement（状態の問い合わせ）が voltage となる。S は問い合わせ先を示し、(プロ

matumot@spring8.or.jp

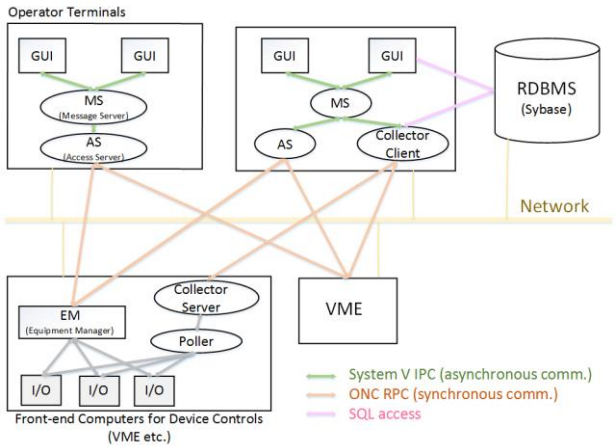

Figure 1: The software structure of MADOCA.

セス番号)_(アプリケーション名)_(ユーザー名)_(ホスト名)から構成され MADOCA 側で割り当てられる。この問い合わせは始めに同一ホスト内の MS(Message Server)がその指令を受け取る。次に AS(Access Server)がメッセージ中の Object 名から宛先を判断し、相手側のホストの EM(Equipment Manager)に指令を送る。EM では機器との応答を行いその結果を取得する。返信時はメッセージが O/V/S/C の形で送られる(相手側の Object は返信時には Subject に入れ替わるため)。C には例えば測定値である”1.5V”がセットされ、GUI では sr_rf_cavity/get/S/1.5V の結果を受け取ることになる。

MADOCA ではこのように SVOC のメッセージ方式により直感的でわかりやすい制御を行うことができる。ユーザーはさまざまな機器の制御を MADOCA の統一的な枠組みを用いて容易に構築することができるメリットがある。しかしながら、長年 MADOCA をさまざまな施設における制御において運用してきた中で、その機能面についていくつかの課題があることが認識されてきた。以下にこれらについて記す。

- SVOC メッセージ長の制限
  - MADOCA では SVOC のメッセージ長は 255 文字までという制限がある。このため、波形データや画像データのような可変長データをメッセージで直接扱うことができない。現在はメッセージ中にファイル名を指定し、そのファイルに可変長データを保存する手段をとっている。しかしながら、間接な手法であるため、余分な処理時間がかかっており、また制御の見通しが悪い状況である。
- 同期型通信による制限
  - MADOCA では制御用端末とフロントエンド計算機のホスト間通信(AS と EM 間)に ONC RPC と呼ばれる Remote Procedure Call を用いている。この通信方法は確実性があり、1 つの送信に対して 1 つの答えが返ってくるように順序化(同期化)されているため扱いやすい。しかしながら、メッセージ送信後に答えが返ってくるまでに次のメッセージを送ることができないため、複数の制御を同時に分散処理させることができない。
- Windows による制御
  - MADOCA は Linux や Solaris での運用を想定して構築された。MS でのプロセス間通信で使われている System V IPC も Unix ライクなシステムでのみ利用できる。このため、ネイティブな Windows 環境上では直接 MADOCA を用いた制御をすることができない。現在は Cygwin を利用する、あるいは socket サーバーを介して外部の計算機と間接的に通信する手法がとられている。
- 制御用端末とフロント計算機間に限定した通信による制限
  - MADOCA では各ホストを制御用端末(クライアント側)か、フロントエンド計算機(サーバー側)かどちらかであるかを事前に規定し、その間で通信を行うことを想定している。このため、制御用端末同士、フロント計算機同士の通信を MADOCA の枠組みで行うことができない。
- Object 管理の必要性
  - AS では Object 名と宛先のホスト名との対応リストを利用しているが、これはデータベースに登録された情報を参照しているため、Object を新規に利用する際には登録手続きが必要となる。また、通常、機器グループ毎(電磁石、RF、真空等)の Object をまとめて扱うように特化された AS をホスト内にいくつか立ち上げて運用していることから、MS が各 Object をどの AS に配信するかを定義する必要がある。このための設定ファイルの管理も必要となり、二重の手間がかかっている。

3. MADOCA II メッセージングにおける機能

これら MADOCA に於けるさまざまな課題を克服するためには、より柔軟なメッセージングを実現する必要がある。様々な検討を行った結果、MADOCA II では ZeroMQ[3]の通信ライブラリを用いてメッセージングソフトウェアを大幅に書き換えた。Figure 2 に示されるように MADOCA II ではメッセージング構成が大きく変更されている。各ホストには MS2 (MADOCA II 用 Message Server)が設置され、任意の 2 つのホストにおける MS2 を経由して通信を行う。3 ホスト間以上でのメッセージのルーティングは行わない。MADOCA II では AS が配置されていな

いが、AS の機能は MS2 の内部に実装されている。以下、MADOCA II で実現された機能について述べる。

- 可変長データへの対応
  - ZeroMQ では、メッセージを複数回に分けて配信することができる。この機能を活かし、通常の SVOC メッセージと同時に画像データなどの付加データを必要に応じて別途送信させることで可変長データへの対応を行った。メッセージ交換に用いるデータは MessagePack[4]を用いてシリアライズ化させ、バイナリデータとして扱っている。MessgePack は多様なデータフォーマットを効率よくデータをコンパクト化させることができるため、我々の用途に適している。また、MessagePack ではバイトオーダーの処理も内部で行われるため、異なる計算機間においてもメッセージ交換を問題なく行うことができる。
- 制御用端末とフロントエンド計算機間の通信の非同期化
  - ZeroMQ では非同期形式の通信がデフォルトであるため、MADOCA で問題となっていた同期通信による制限をなくすことができる。非同期制御をより効果的に行うため、EM 部を必要に応じてデバイス毎に分割させることができる設計にした。EM を分割させることにより複数の制御を分散化させ高速処理をさせることができる。他には、特定のデバイスが故障した場合においても残りのデバイスで運用させることで運転の継続性を向上させるメリットもある。これら非同期制御の利用では、複数のメッセージを一度に送信し、後でメッセージをまとめて受信する場合が想定される。送信と受信間のメッセージの対応関係をつけるため、各メッセージにはメッセージ番号を割り当て、メッセージ受信時にメッセージ番号を指定することができるようにした。
- Windows による制御
  - MADOCA II では開発当初からマルチプラットフォームでの利用を想定してソフトウェアの構築を行っている。ZeroMQ 及び MessagePack もマルチプラットフォーム対応である。MADOCA II メッセージングのプログラムは C++で書かれており、Windows 上においても同一のソースプログラムを Visual C++を用いてビルドすることで利用が可能である。
- 制御用端末間、およびフロントエンド計算機間のメッセージ通信
  - MADOCA II では MADOCA と異なり、各ホストは、必要に応じて、クライアント、またはサーバーとしての機能をもたせることができる。したがって制御用端末とフロントエンド計算機間の通信のみでなく、制御用端末間、フロントエンド計算機間の通信も MADOCA II の枠組みで同様に行うことができる。例として Figure 2 では制御用端末における MS2 間を接続させ、相互に通信できるようにした例を示している。
- Object 管理の自動化
  - MADOCA II では各 Object の配信先のリストが MS2 内に自動登録されており、この情報を利用してメッセージ配信を行うことができる。これは、各 EM が扱う Object リストを、EM 起動時に同一ホスト上の MS2 に登録することで実現している。登録された Object 情報は接続されているホスト間における MS2 にも伝播される。このように MADOCA II では Object 管理が自動化されているため、データベースによる手続きが不要となり管理の手間を省くことができる。なお、Object 名のユニーク性は MADOCA と同様 MADOCA II においても要求されているため、Object の２重登録を行うことはできない。

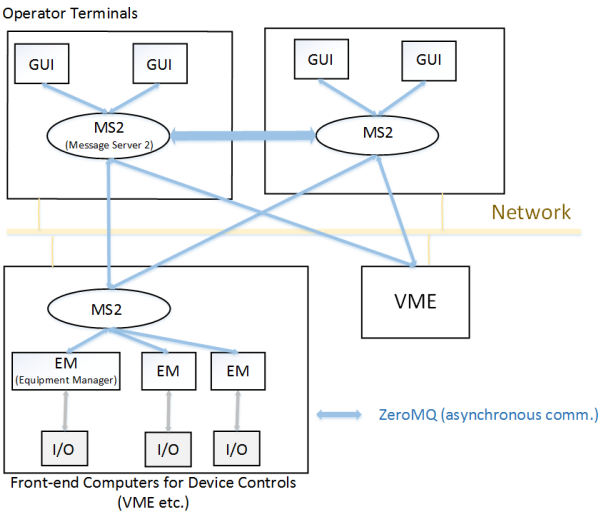

Figure 2: The software structure of MADOCA II messaging.

4. MADOCA II の利用例

今回開発した MADOCA II メッセージングは SPring-8 の制御系においていくつかの事例で導入されてきており、MADOCAII で新しく拡張された機能の有用性が示されている。これらの内容については本学会資料[5][6]を参照されたい。どのアプリケーションでも波形データ、画像データなどの可変長データが MADOCA II の枠組みを用いて制御で直接取り扱われている。[5]の例では、Windows 上において MADOCA II の制御系が構築されており、Windows 環境でのみ制御可能な機器の制御が

MADOCA II-LabVIEW インターフェースを用いて実装されている。

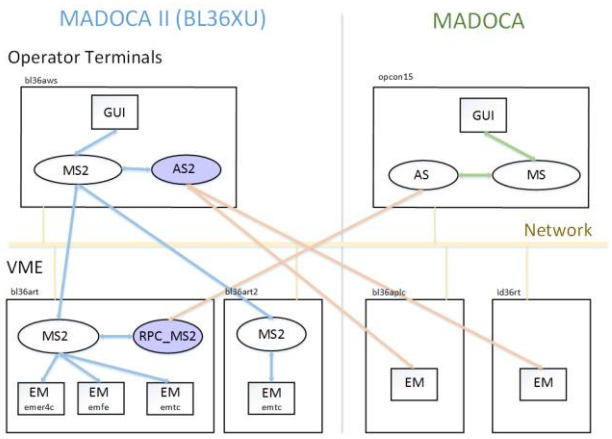

Figure 3: An example of the implementation of MADOCA II messaging at BL36XU. AS2 and RPC_MS2 interfaces are used to connect controls between MADOCA and MADOCA II.

5. MADOCA II 制御系への移行について

現在、我々は SPring-8 の加速器およびビームラインの制御系を MADOCA II に移行させる計画を進めている。今まで MADOCA で構築されてきた多くのアプリケーションを MADOCA II にスムーズに移行させるため、MADOCA II で用意した関数は MADOCA の関数と後方互換性を持たせている。よって基本的には再コンパイル、再リンクをするのみで MADOCA から MADOCA II への移行できる。

MADOCA II を実環境で本格運用するため、昨年夏から BL36XU において MADOCA II の試験導入を行った。このことにより、MADOCA II の運用時の安定性について問題がないことを確認するとともに、制御トラブル時の対処方法も確立することができた。BL36XU における MADOCA II 実装については Figure 3 に示される。BL35XU では MADOCA II が導入されるが、他で稼働している MADOCA 制御系とも共存させる必要がある。このため、MADOCA と MADOCA II 間を接続させる 2 つの制御インターフェースを整備した。1 つは、MADOCA II 用ホストから MADOCA 用ホスト を制御する際に用いる AS2 (MADOCA II 用 Access Server) であり、他は MADOCA 用ホストから MADOCA II 用ホストを制御する際に用いる RPC_MS2 (MADOCA 側の AS と MADOCA2 側の MS2 の仲介エージェント) である。どちらのインターフェースも MADOCA II 側のホスト上で稼動させる。

制御系全体の MADOCA II への移行については、制御用計算機(GUI 側)の MADOCA II 移行を先に進める方向で検討している。これは、制御系において制御用端末の数のほうが少ないこと、運転制御に用いる制御用端末は、混乱をさせるために同一の制御方式に速やかに移行させた方が好ましいこと等が理由として挙げられる。SACLA の実験ステーションにおけるデータ収集系における制御に関しては、この夏の停止期間中に制御用端末側を全て MADOCA II に置き換える予定である。SPring-8 加速器・ビームライン制御系に関しては、来年の春をターゲットに制御用端末側の MADOCA II 移行の準備を進めている。

一旦制御用端末側を MADOCA II に移行させた後は、フロントエンド計算機を MADOCA II へ適宜移行させていくことで全体の制御系を MADOCA II で置き換えることができる。フロントエンド計算機側では多様な機器を制御するために、Solaris、Linux、ARM 計算機等さまざまな OS が機器制御に利用されており、これら全ての実行環境において MADOCA II を稼動させる必要がある。MessagePack では gcc4.1 以上のコンパイル環境が必須となるため、古いシステムでは今まで利用してきた開発環境を利用することができない場合もあった。このため、必要に応じて MADOCA II 専用の開発環境を整備することで対応を行っている。

6. まとめ

MADOCA の制御フレームワークを ZeroMQ の通信ライブラリを用いて大幅に書き換え、機能を拡張した新しい制御フレームワーク MADOCA II を開発した。既に、波形データ、画像データなどの可変長データの利用、Windows 環境での制御において MADOCA II が利用されてきており、今後、SPring-8 加速器の高度化など、さまざまな側面で MADOCA II が活用されていくことが期待される。

謝辞

MADOCA II メッセージングの開発を進めるにあたり、有益な討論、ご意見を頂いた(公財)高輝度光科学研究センター 田中良太郎氏、山下明広氏、籠正裕氏、他制御・情報グループの方々に深く感謝致します。

参考文献

[1] R. Tanaka et al., “Control System of the SPring-8 Storage Ring”, Proceedings of ICALEPCS’95, Chicago, p.201(1995)
[2] 山下明広,他 “MADOCA II データ収集と蓄積システム”,本プロシーディング,MOOS09
[3] http://www/zeromq.org/
[4] http://msgpack.org/
[5] 古川行人他,”MADOCA II-LabVIEW インターフェースを用いた BPM データ読み出しシステム”,本プロシーディング,MOOS10
[6] 清道明男他,”SPring-8 蓄積リング 2 次元放射光干渉系計高度化に向けた MicroTCA 画像処理システムの開発”, 本プロシーディング, MOOS11