MADOCAII データ収集・蓄積システムの現状

STATUS OF MADOCAII DATA ACQUISITION AND STORAGE SYSTEM

山下明広、籠正裕
Akihiro Yamashita*, Masahiro Kago
Japan Synchrotron Research Institute/SPring-8

Abstract

The current status of MADOCA II data acquisition system is described. We developed high performance, flexible, reliable and easily maintainable MADOCA II system for advanced accelerators and beam-line control systems. We began to install MADOCA II to SPring-8 control system. We decided to run MADOCA II data acquisition system and old MADOCA in parallel until MADOCA II replaces MADOCA completely. Data acquisition system, data access API, alarm system and web system are developed in MADOCA II environment. MADOCA II also shows its advantages in fast COD data acquisition and trouble detection of magnet power supplies.

1. はじめに

SPring-8 では 1997 年のコミッショニング以来、加速器とビームライン制御フレームワークとして MADOCA(Message And Database Oriented Control Architecture) [1] を使用している。MADOCA はその名前が示すとおり、データの管理をデータベースに依存しており、運転パラメータを管理し、ログデータを常時記録することで、SPring-8 の安定運転と高度化に寄与してきた。現状ではリレーショナルデータベース管理システム(RDBMS)の Sybase を利用したシステムで運用しているが、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を考慮すると、システム拡張性、データ収集に対する柔軟性やメンテナンス性などの課題が存在する。

これらを改善するために、次世代の加速器制御フレームワーク MADOCAII データ収集・蓄積システムは開発された [2] [3]。Figure 1 に示すように、新システムはデータ収集の通信に ZeroMQ [4] と MessagePack [5]、データ蓄積部に NoSQL(Not only SQL) データベースである Redis [6] と Apache Cassandra [7]を使用して構築され、現行 MADOCA と比較して高信頼性、高性能、高拡張性かつ柔軟なデータ管理の実現がテスト環境での実証試験により証明されてきた。

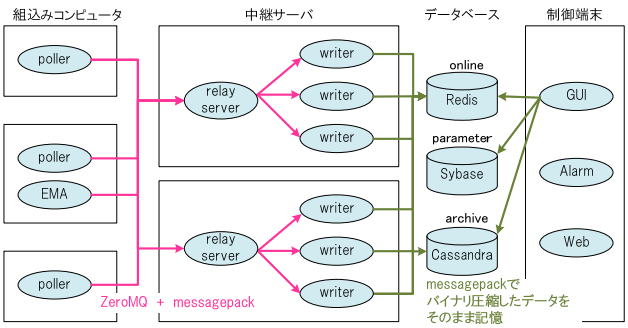

Figure 1: System architecture of MADOCAII.

この結果を踏まえ、我々は本格導入に向けた移行準備を始めた。SPring-8 の加速器およびビームラインのデータ収集系を新システムへ移行するためには、データベースはもとより、それにアクセスしているアプリケーション類を移植する必要がある。加えて、データ収集プロセスが稼働する組込みコンピュータは、SPring-8 全体で 370 台にも及び、これら全ての実行環境において新しいデータ収集系を稼働させなければならない。そこで我々は、移行時のリスクおよび移行失敗に伴う回復の容易さを考慮して、以下の方針で移行を計画している。

- 移行は段階移行方式とする。まず初めにデータベース、その蓄積データを利用する制御 GUI、Web アプリケーション、アラームシステムなどの上位系を移行する。
- 上位系移行が完了し安定運用を開始した後、下位系であるデータ収集系を順次移行する。
- 上位系及び下位系の完全移行が完了するまで、新システムと現在の RDBMS を利用したデータ収集系は並行運用を行う

本稿では、この移行計画に基づき実施した取り組みについて述べる。また、実機投入した MADOCAII データ蓄積システムは既にいくつかの事例で SPring-8 制御系に導入されているため、これらについても紹介する。

2. システム構成

本格導入に向けた準備として、新システムを実環境の制御系へ構築した。システム構成を Figure 2、ハードウェアやソフトウェアの構成品目情報を Table1 に示す。本システムはログデータの永続保存用データベースとして 6 台のサーバで Cassandra クラスタを構成している。クラスタ内ノード間で通信のため、専用のネットワークを使用した。2 台のサーバをデータ転送処理やデータ書込みを行う中継サーバに割り当て、これらを Gigabit Ehternet で加速器制御系ネットワークに接続している。

aki@spring8.or.jp

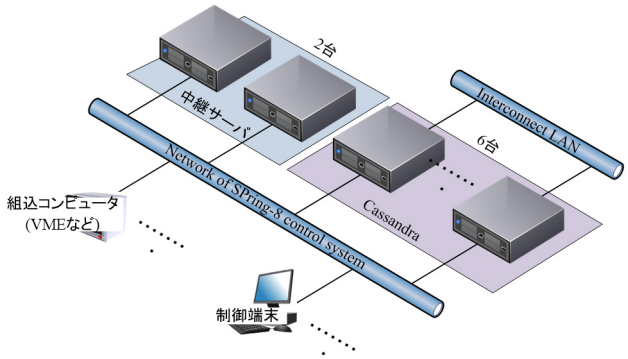

Figure 2: Computers and network structure.

Table 1: Specifications of Servers

Role	Composition
Cassandra server	Dell PowerEdge R420 OS: CentOS 6.4 (64bit) CPU: Intel Xeon E5-2430, 12core, 2.2GHz MEM : 16GB HDD: 600GB SAS 15Kr/m x1 3TB SATA 7,200r/m x3 Cassandra version : 1.2.13 Java : JRE1.6.0_45
Relay server	Dell PowerEdge R420 OS: CentOS 6.4 (64bit) CPU: Intel Xeon E5-2430, 12core, 2.2GHz MEM : 16GB HDD: 600GB SAS 15Kr/m x2 (RAID1) Redis version : 2.6

Cassandra が動作するサーバは OS 等のシステム領域確保のための 600GB HDD を 1 基、データ蓄積のための 3TB HDD を 3 基搭載している。Cassandra バージョン 1.2 以降から導入された JBOD (Just a bunch of disks) 機能を使用しているため、HDD は non-RAID 構成とした。本機能の利用により、I/O リソースが確保し易く、Cassandra を停止させることなくディスクのホットスワップが可能になるなどのメリットもある。Cassandra クラスタ内ではデータは 3 重化され、データの可用性を向上させている。

中継サーバ上では、最新値のみを保持するインメモリ型データベースの Redis、メッセージを仲介する Relaysever, メッセージをデータベースに格納する Writer×20 プロセスが動作する。これは 1 台の中継サーバで現在 SPring-8 にて収集している信号点数を扱うことができる性能を持つ。しかしながら、可用性や信頼性を考慮して、中継サーバを 2 台構成として冗長化を図った。

3. 上位系移行

3.1 データ移行

SPring-8 運転開始 (1997 年) 以来、Sybase データベースに蓄積されている約 4TB のデータを Cassandra へ移行した。移行後のデータ容量は、約 3TB/node であった。また、データは圧縮率こそ小さいものの、圧縮・解凍の時間が極めて短い SNAPPY 形式の圧縮を行って Cassandra に格納している。

3.2 データ収集

新システムと旧システムの収集系を並行運用するための移行ツールとして、現 MADOCA の持つ機能を利用したデータ収集系を整備した。Figure 3 に構成を示す。

現 MADOCA におけるデータ収集プロセスはポーリング型であり、制御端末上で動作する Collector Client(CC) と、組み込みコンピュータ上で動作する Poller/Collector Server(CS)から構成される。Poller は周期的に機器からデータを取得して共有メモリへ書き出し、CS は CC から定期的に送られてくるデータ収集命令に従って、この共有メモリ上のデータを収集している

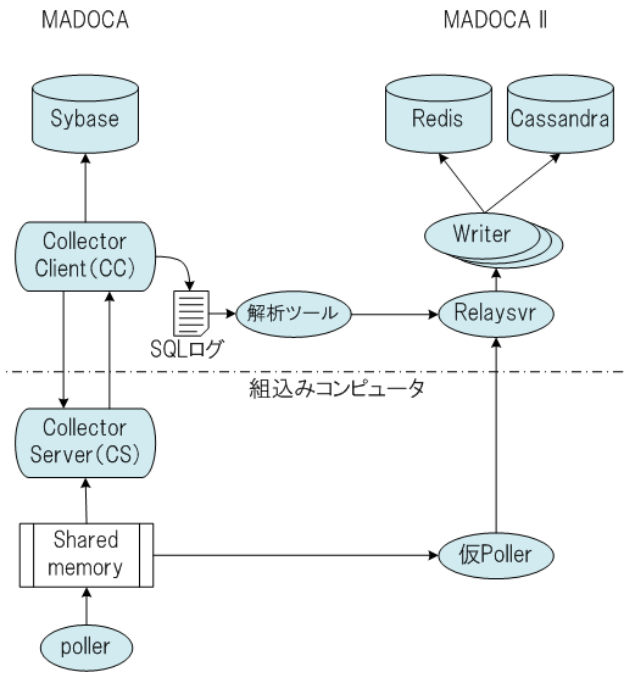

Figure 3: Data acquisition system. MADOCA and MADOCA II are running in parallel.

MADOCAII では 2 種類の書き込みプロセスを作製して動作させた。1 つは組み込みコンピュータ内で動作し、Poller プロセスがシェアードメモリに書いたデータを読んでそれを独自周期で中継サーバに送る。これは組込みコンピュータ内に別プロセスを動作させることになり資源(CPU,メモリ)に余裕がある場合に使用できる。このプロセスは C++で実装した。

もう 1 つは、組み込みコンピュータの資源に余裕が無い場合のために CC の SQL 出力機能を使用するものである。CC には、RDB に書込む SQL 文を標準出力に出力する機能がある。標準出力はファイルに書込まれる。そのファイルの更新された部分の SQL を解釈し、MADOCAII データ収集メッセージに変換し中継サーバに送るプロセスを作製した。このプロセスは Python (v2.6)で実装した。

ここでファイル出力を利用するために 2 つの問題があった。1 つは出力ファイルサイズが増大しファイルシステムが満杯になってしまうことであり、もう 1 つは標準出力のバッファリングのため、リアルタイムのデータが得られないことである。最初の問題は Apapche rotatelogs ユーティリティで解決した。これは指定された時間またはサイズでログファイルを交換するのでファイルシステムを満す心配が無くなった。また unbuffer コマンドは標準出力をほぼリアルタイムでファイルに書込む。

以上の 2 つの方法により、現行の MADOCA と MADOCAII が互いに干渉することなく並行にデータ収集を行うことができた。

これら収集系は、2014 年 1 月から現行と同じ約 27000 点以上のデータを周期的に収集し、安定稼働を継続している。

3.3 データベース API の整備

制御系 GUI アプリケーションが使用するためのデータベース API(C/C++言語対応)を整備した。MADOCAII では最新値を Redis で保持し、アーカイブデータは Cassandra で管理している。本 API を使用すればそれらデータベースを意識することなく、必要なデータを取得することができる。また、API の関数名、引数、戻り値などは現行 MADOCA が提供している API と互換性を持つよう開発しているため、GUI アプリケーションはソースコードを改変することなく再リンクのみで動作する。

本番運用に向けて、全ての GUI の再コンパイルを行った。コンパイルした全 GUI は、約 1 か月の動作検証を行い、正常動作を確認した。さらにマシンスタディ時に一部の GUI を実際に動作させ、加速器運転のオペレーションに使用しても問題がないことを確認した。

3.4 Web サービス

現在の Web システムのうちログデータ収集に関係するものは機器のサブグループ毎に一括で現在値を表で表示するページと各信号(複数信号を重ね書きすることもある)時系列データをグラフまたはデータダンプするページがある。Web サーバは Apache でその中から Python で実装された CGI プログラムを呼びだし、Gnuplot の画像を表示する。その機能と外観を変更することなく、オンラインデータは Redis から、時系列データは Cassandra から取得するページを新たに作製した。

しかし、これらのページの外観は現代の標準 Web ページと比較して、あまりに古色蒼然としている印象を免れない。また機能的にも劣る。そこで現在以下のような新 Web ページを作製中である(Fig.4)。

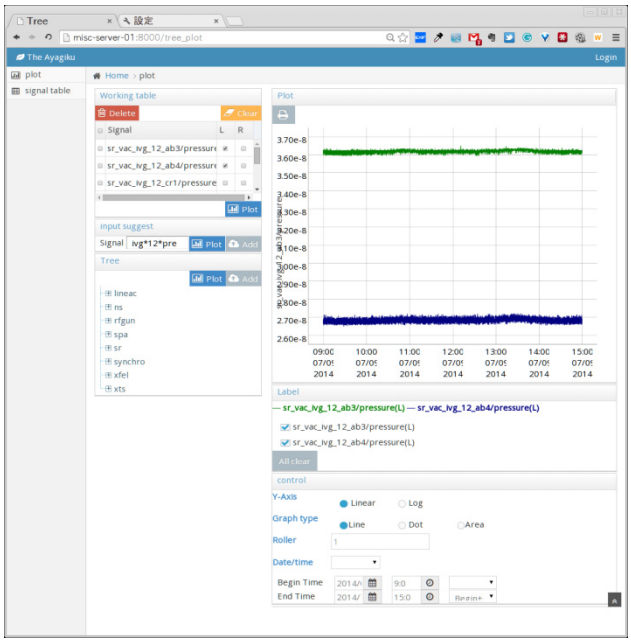

Figure 4: Example of data display web page.

Twitter Bootstrap v3 [8]を使用することによってグリッドレイアウトでレスポンシブ(ブラウザの大きさに応じて表示内容を最適な状態に変化させる)な外観を実現する。デザインについては市販の Bootstrap 用テンプレートを使用することにより、素人でもデザイン性にすぐれたサイト構築を可能にした。

グラフ表示についてはブラウザで Javascript を使用して時系列グラフを表示するライブラリは多く出回っている。代表的なライブラリを候補に高速性、インタラクティブな機能や時系列データ表示などの要求を満すものとして dygraphs [9] ライブラリを採用した。

web サーバは、テンプレートの使用のしやすさ、非同期 I/O による高速性、websocket のライブラリなどの要素から Tornado Web [10]サーバを採用した。また Cassabdra は並列化により読み出しが高速化されるので、これにも対応した。

3.5 アラーム監視

MADOCA では RDB を全面的に使用したアラームシステムを使用している。これは、RDB から閾値など標準値を読み込み、RDB に書かれたリアルタイムデータと比較し、異常があればその情報を RDB に書込む。

各信号などのデータと閾値や異常データを関連付けて管理することは RDBMS の得意とするところであって、この機能を NoSQL で実現することは難しい。MADOCAII では RDB と NoSQL のそれぞれの利点を活用することが基本方針であるため MADOCA アラームのうち、リアルタイムデータの読み込み部分のみを NoSQL の Redis に移行することとした。

実装は Python (v.2.6)でおこない、GUI は Qt4 [11], PyQt4 [12]を使用した。GUI デザインは Qt designer を使用し、GUI インターフェイス部分とアラームアルゴリズム部分を分離してコーディングできるようにした。

アラームアルゴリズム部分は、GUI 無しでも起動できデバッグを容易にしている。MADOCA アラームでは、機器グループ(例:SR 真空、SR 電磁石等)毎にアラームプロセスを起動するという形式だったが、機器グループ毎の GUI アプリケーションを部品化することにより、機器グループ GUI アプリケーション単独でも、それらをまとめた GUI アプリケーションでも動作できるようにした。

アラームシステムは 2014 年始めから、現在のアラームシステムと並行で運用をおこない、アラームの動作タイミングに起因する違い以外は同様な動作をすることが確認された

4. 新システムの利用事例

今回実環境に整備した MADOCAII データ収集・蓄積システムは SPring-8 の制御系において poller/collector 系以外でもいくつかの事例で既に導入されてきており、現行の MADOCA では難しかった事例で新システムの有用性が示されている。

4.1 蓄積リング COD データの蓄積

蓄積リング COD の構造化されたデータの 10Hz 蓄積に新システムを使用している [13]。旧 MADOCA システムでは RDBMS の書き込み性能上の問題から 0.1Hz 蓄積にとどめてきたが、新システムを用いることで 10Hz 蓄積が可能になった。構造化されたデータは MessagePack を用いてシリアライズ化してデータベースに保存しているため、読み込み性能にも優れる。また、10Hz で収集されたデータは 1 週間経過の後、0.1Hz に間引いてデータを永続的に保存する仕組みとなっている。これは Cassandra に標準で組み込まれている TTL(Time To Live)機能を用いて実現している。これにより蓄積データの肥大化を避け、有限なディスクの効率化を図った。

4.2 蓄積リング ステアリング電源の異常波形蓄積

蓄積リングステアリング電磁石電源では時折スパイク状の異常電圧が発生するという現象が見られてきた。これは故障の前兆であり、確実な捕捉が要求される。MADOCAII により異常を検知した時に、異常前後のデータを蓄積できるようになった。この異常データの検知と蓄積により電源故障の未然の防止と故障解析ができるようになった。このような梶泰之イベントトリガによるデータを収集は MADOCA ではサポートされていなかったがユーザからニーズがあった。新フレームワークを用いて本システムが稼働したことで、有用性が実証され、また、ユーザのニーズを満たし今後の加速器高度化を支える基盤ができあがった。

5. まとめ

MADOCAII データ収集・蓄積システムを実環境上に構築して SPring-8 運転開始以来蓄積してきたデータの移行、GUI、Web、アラームシステムなどデータベースにアクセスするアプリケーション類の整備を行った。これにより本格導入に向けた上位系移行の準備が整った。そこでこの夏、移行プランに従って上位系を移行し、2014 年秋以降の運転からの実運用を検討している。また、既にいくつかの加速器制御システムにおいて新フレームワークの持つ特徴が有効活用されてきており、今後の加速器の高度化など、さまざまな面で MADOCA II データ収集・蓄積システムが活用されていくことが期待される。

謝辞

本研究にあたり、田中良太郎, 古川行人, 石井美保, 松本崇博, 酒井久伸の皆様には貴重なご助言をいただきました。植田倉六さん、梶泰之さんには制作やテストでご協力をいただきました。ここに感謝を記します。

参考文献

[1] R. Tanaka. et al., “The first operation of control system at SPring-8 storage ring,” *ICALEPCS 1997*, Beijing, China, 1997.

[2] A.Yamashita et al., “MADOCA II データー収集と蓄積システム,” 第 10 回加速器学会年会,2013.

[3] M.Kago et al., “Development of A Scalable and Flexible Data Logging System Using NoSQL Databases,” *ICALEPCS 2013*, San Francisco, USA, 2013.

[4] http://www.zeromq.org.

[5] http://www.msgpack.org.

[6] http://redis.io.

[7] http://cassandra.apache.org.

[8] http://qt-project.org.

[9] http://www.riverbankcomputing.co.uk.

[10] http://getbootstrap.com.

[11] http://dygraphs.com.

[12] http://www.tornadoweb.org.

[13] T. Fujita et.al., “SPring-8 蓄積リング COD データ取得の高速化,”第 1 1 回加速器学会年会, 2014.