MADOCA II データ収集・蓄積システムの SPring-8 制御系への実装

IMPLEMENTATION OF MADOCA II DATA ACQUISITION AND STORAGE SYSTEM AT SPRING-8

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

Abstract

The data acquisition and storage system for SPring-8 accelerator control was upgraded to MADOCA II. It has been collecting all the log data required for accelerator control without any trouble since the upgrade. In the new system, we adopted two NoSQL databases, Apache Cassandra and Redis, for data storage. Data acquisition part of the new system was built based on ZeroMQ message packed by MessagePack. The operation of the new system started in January 2015 after the long term evaluation over one year. In this paper, we will discuss the implementation of MADOCA II system into SPring-8 and operation status.

1. はじめに

SPring-8 加速器制御のためのデータ収集・蓄積システムを従来の MADOCA(Message And Database Oriented Control Architecture)[1] から MADOCA II に更新した。運用開始後は大きなトラブルもなく順調に稼働している。

従来の MADOCA では単一のリレーショナルデータベース管理システム(RDBMS)を用いて、加速器の機器情報、運転パラメータ、およびログデータを一元管理してきた[2]。1997 年のコミッショニング以来、SPring-8 の安定運転に寄与し、加速器高度化と共に発展、進化してきた。しかしながら、近年の加速器高度化や今後の SPring-8 II で想定され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あった。

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[3][4]。これまでの MADOCA 運用経験と近年の技術的進歩を新システムに取り入れており、単一障害点がなく可用性に優れ、スケーラブルに大量データを扱うことができるなどの特徴を持つ。他にも、データベースの内部構造を改変することなく様々なデータ型に対応することができ、従来の MADOCA では実現困難であったことが可能となる。

昨年度の学会で報告したとおり、新システム本格運用の開始に先立って、実環境下にプロトタイプを先行導入し、約 1 年にわたり最終評価および運用経験を積んだ[5]。この導入において、新システムは安定稼働を続け大きなトラブルもほとんどなく、システムの健全性や整備したアプリケーション類に問題ないことが実証された。しかしながら、システムをより安定に運用するためのいくつかの知見が得られた。この結果を反映させた上で、2015 年 1 月から SPring-8 制御系のデータ収集・蓄積システムを MADCOA から MADOCAII に置き換え、新システムを用いた加速器運転を本格的に開始した。

kago@spring8.or.jp

2. SPring-8 制御系への実装

SPring-8 制御系へ実装した全体構成を Figure 1 に示す。当初の計画どおり MADOCAII への移行は段階移行方式とし、データベースの蓄積データを利用する制御 GUI、Web システム、アラームシステムなどの上位系を移行した。

2.1 システムの概略

新システムでは、データ収集の通信に ZeroMQ[6] と MessagePack[7]、データ蓄積部に NoSQL データベースである Redis[8]と Apache Cassandra[9]を使用した。このシステムにおけるデータ収集では、従来システムとの並行運用を実現するために、フロントエンド計算機内で動作する po2m2db と旧 MADOCA 収集系のログ出力を利用する cc2m2db の 2 種類の書込みプロセスを動作させている。これらは、データを非同期、一方通行のメッセージの形で中継プロセス(relaysvr)に送る。メッセージは ZeroMQ によって自動的にロードバランシングされ中継プロセスに分散される。中継プロセスは、対障害性のために複数動作させており、po2m2db と cc2m2db からのソケット数の管理、データベース書込みプロセス(writer)へのメッセージ転送の役割を担う。データベース書込みプロセスは、メッセージを加工し Redis(データキャッシュ)と Apache Cassandra(アーカイブストレージ)に専用スレッドから同時に書き込む。複数の書込みプロセスを動作させることで、書込処理を分散してデータベースへの書込みパフォーマンスを向上させるとともに、中継プロセスから送られるメッセージの滞留を防いでいる。

ログデータは用途に応じた複数データベースに格納されるが、整備した専用 API(C/C++言語対応)を用いることで、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、Web システムを用いれば、プログラミングすることなくデータ表示やグラフ表示、データダンプすることができる。

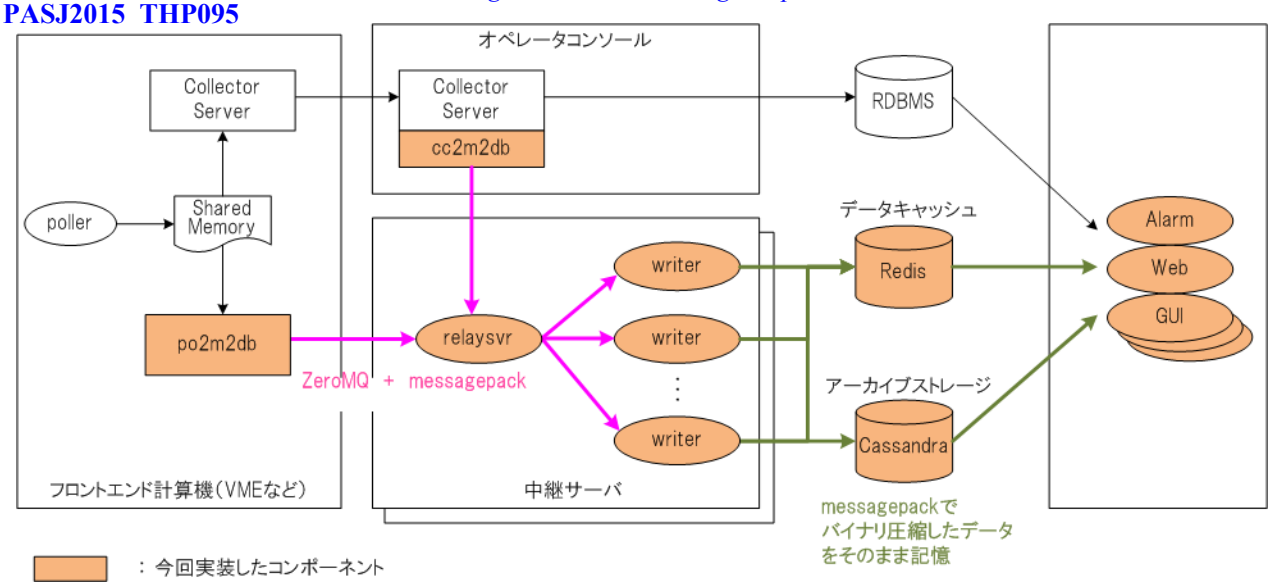

Figure 1: System architecture of MADOCA II.

2.2 導入にあたり

先行導入したシステムの運用経験を元に、より安定かつ効率的に Apache Cassandra を運用するために構成の見直しを行った。

2.2.1 テーブル構造の変更

先行導入したシステムでは単一テーブルに全てのデータを格納するテーブル構造であったが、この構造は Cassandra のデータ管理メカニズムに適さないことが分かった。Cassandra ではテーブル毎にデータを管理しており、SSTable と呼ばれる複数のファイルで保存される。これらファイルはコンパクションという処理によって次第に集約される。このコンパクション処理はディスク容量が許す限り繰り返されるため、全データを 1 つのテーブルで管理すると SSTable が肥大化する現象を招く。ファイルが肥大化するとメンテナンス性が損なわれるだけでなく、コンパクション実行中は高負荷状態となりパフォーマンスにも影響を及ぼす可能性があった。また、コンパクションの過程では最大で処理対象テーブルと同じ空き容量が要求されるため、データを保持できる容量はディスクの半分程度しか使用することができず非効率であった。

そこで、本番導入したシステムでは、月毎にテーブルを分割する構造に変更した。1 つのテーブルが持つデータ容量を小さくし、コンパクション対象となるデータに制限を設けることで、ファイルの肥大化を避ける。これにより、負荷の高いコンパクション処理の時間短縮、ディスクの有効活用、バックアップやリストアが容易になるなどの利点が生まれる。デメリットとしては、月をまたぐようなデータ取得の場合、複数のテーブルに対してアクセスが必要となるが、これについては API 関数を改修して対応した。

2.2.2 クラスタの拡張

先行導入したシステムは 6 台のサーバで Cassandra クラスタを構築していたが、扱うデータ量に対してクラスタが小規模であることが判明したため、12 台クラスタへと拡張した。拡張においては、テーブル構造の変更も行う必要があったため、新たにサーバを購入し別クラスタを構築した後、実機環境へインストールした。同時に、Cassandra のバージョンを 1.2 系から仮想ノード(vnode)の作成機能がデフォルトになった 2.0 系へアップした。この vnode 機能は 1 つのマシン上に複数の仮想ノードを作成でき、他サーバのレプリカから高速にデータリペアができるなどの特徴を持つ。これを利用することでクラスタ管理を効率化できるほか、サーバに障害が発生した際の復旧も容易になり、全体的なパフォーマンスも向上する。Cassandra に用いたハードウェアやソフトウェアの構成品目情報を Table 1 に示す。

Table 1: Specifications of Servers

Role	Composition
Cassandra server	Dell PowerEdge R420 OS : CentOS 6.6 (64bit) CPU : Intel Xeon E5-2420 v2, 6c, 2.2GHz MEM : 16GB HDD : 600GB SAS 15Kr/m x1 3TB SATA 7,200r/m x3 Cassandra version : 2.0.10 Java : JRE1.7.0_67-b01

3.3 データ移行

テーブル構造変更およびクラスタ拡張に伴い、SPring-8 運転開始 (1997 年) 以来、RDBMS に蓄積された約 4TB のデータを Cassandra へ移行した。移行後のデータ容量は、約 750GB/node であった。

3. システム診断ツール

MADOCAII データ収集・蓄積システムの安定運用を担保するため、各種診断ツール類を整備した。

3.1 リソース監視

本システムでは Zabbix[10]を用いて各サーバの各種リソースを一元的に監視できるようにした。Zabbix は Web ブラウザ上の操作により監視ステータスの表示や設定を行うことができる。さらに Zabbix 自身に JMX 監視機能が組み込まれているため、Cassandra の様な Java アプリケーションの監視が容易に実現できた。現在は、CPU ロードやメモリ使用量など、Linux サーバで一般的なサービスの監視に加え、Cassandra が使用する Java ヒープ状態やタスク情報を監視対象としている。これら情報を元にシステム健全性の把握はもちろん、今後のサーバやプロセス増設の計画に役立てる。

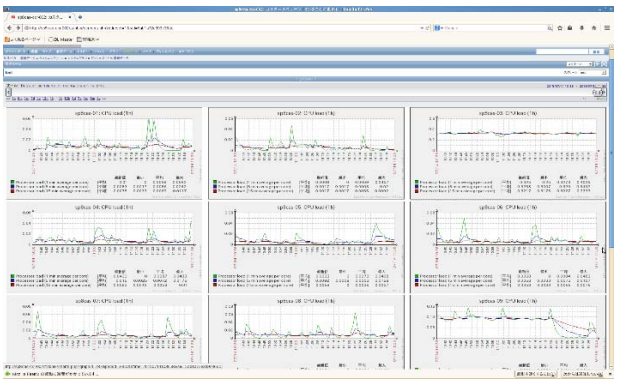

Figure 2: Resource monitoring by Zabbix.

3.2 プロセス内部状態の監視

中継プロセス、データベース書込みプロセスの内部情報を表示するツールを整備した。これは Qt[11]で作成した。本ツールでは、1 秒間に処理したメッセージ数やデータベース書込みに要した時間などプロセス内部情報の表示に加え、簡単なメッセージ送受信テストも行うことができる。本ツールはシステム管理者用として制作しており、トラブルシュートやプロセス動作状況の監視に用いる。

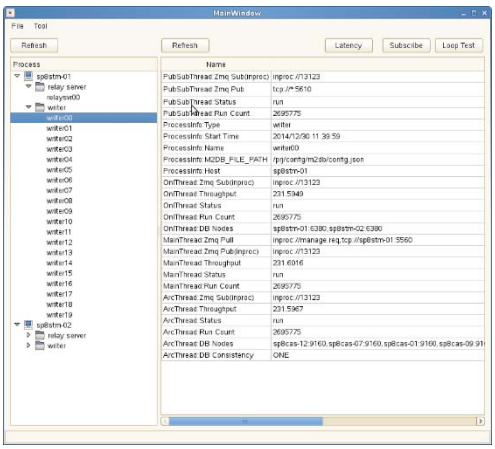

Figure 3: GUI for monitoring MADOCA II process.

そのほか、ユーザ用として各種プロセスの動作状態をモニタする GUI も用意した。これは中央制御室の中央に位置する大型ディスプレイ上で動作し、一目でシステムが正常動作しているかどうかが判断できる。

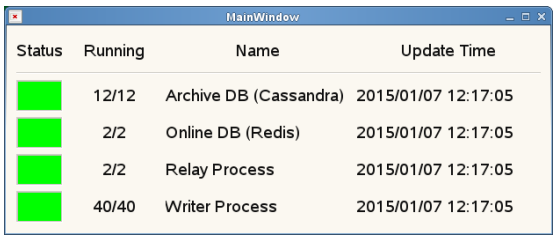

Figure 4: A system monitor of MADOCA II.

3.3 異常検知

本システムは単一障害点のない高い可用性を持っているが、トラブルを早期に検出しリカバリ作業を速やかに行うために、プロセスの異常検知で管理者にメール通知する仕組みを Python で実装した。健全性チェックのため、異常の有無に関わらず 1 日 1 回巡回結果をメール通知する機能も実装した。

4. 現在の状況と今後

2015 年 1 月から MADOCAII データ収集・蓄積システムを用いた加速器運転を本格的に開始した。事前に十分な時間をかけ動作確認を行った上で導入したため、移行に伴う大きなトラブルもなく安定した動作を行っている。現在のところ、従来の MADOCA と同じ約 27,000 点の信号を扱い、秒間あたり約 9,000 メッセージを蓄積している。また、各サーバの CPU ロードやメモリ、HDD 使用率などのリソース状況にも問題は見られていない。

今回移行を終えたのは上位系のみであり、フロントエンド側で動作する収集プロセスは従来の MADOCA の仕組みを利用したものである。フロントエンド計算機は、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るべく、現在整備を進めている。同時に本システムに対する信号登録手順の見直しも行っている。旧来のシステムでは、組込コンピュータでの設定ファイルと、データベース登録ファイルが分離しており、これが信号登録の簡便化にとって障害となっていた。見直し後はこれを一本化し簡便化と誤りの減少を図る。

5. まとめ

MADOCAII データ収集・蓄積システムの本格運用を開始した。システム構築にあたり、昨年度の先行導入の運用経験を反映させ、Apache Cassandra のテーブル構造変更、クラスタ拡張、並びにシステム診断ツール類を整備した。このシステム導入に伴い、蓄積データを利用する制御 GUI、Web システム、ア

ラームシステムなどの上位系を移行した。現在のところ、大きなトラブルもなく順調に稼働している。今後はログデータを生成する下位系の整備を行うとともに、信号登録方法の見直しも行い、MADOCAII完全移行を目指す。

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., "The database system for the SPring-8 storage ring control," *ICALEPCS 1997*, Beijing, China, 1997.
[3] 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.
[4] M.Kago et al., "Development of A Scalable and Flexible Data Logging SYstem Using NoSQL Databases," *ICALEPCS 2013*, San Francisco, USA, 2013.
[5] A.Yamashita et al., "MADOCA II データー収集・蓄積システムの現状," 第 11 回加速器学会年会,2014.
[6] http://www.zeromq.org
[7] http://www.msgpack.org
[8] http://redis.io
[9] http://cassandra.apache.org
[10] http://www.zabbix.com
[11] http://qt-project.org