

Hadoop・HBase を利用した J-PARC 運転データアーカイビングの現状

STATUS OF J-PARC OPERATION DATA ARCHIVING USING Hadoop AND HBase

菊澤信宏^{#, A)}, 吉位明伸^{A)}, 池田浩^{A)}, 加藤裕子^{A)}

Nobuhiro Kikuzawa^{#, A)}, Akinobu Yoshii^{A)}, Hiroshi Ikeda^{A)}, Yuko Kato^{A)}

^{A)} Japan Atomic Energy Agency

Abstract

J-PARC (Japan Proton Accelerator Research Complex) is controlled with a lot of equipment, and we archive the operation data of about 64000 EPICS records in Linac and RCS. PostgreSQL is used in the data archiving system now, but it has a problem of extendibility and data migration and so on. In order to cope with the problem, we have examined the next-generation archive system using Hadoop of a distributed processing framework and HBase of distributed database. We developed the test system and use it in a tentative way.

It is mentioned that a master node is SPOF (single point of failure) as a problem of Hadoop and HBase. So, we make two servers into HA (High Availability) cluster structure using Heartbeat and Pacemaker, and raise system availability. Moreover, important metadata for data management is protected by the data replication between two servers using DRBD (Distributed Replicated Block Device). About 50 TB of HDFS (Hadoop Distributed File System) is built using nine slave nodes, and HBase is worked on it. About 6,500 polling type data in a cycle of 1-60 seconds and about 13,000 the event type data in Linac are collected in a tentative way now, and it can be archiving stably. Moreover, about data retrieval, there is a case where response time is shortened by about 1/5 as compared with the present system.

This paper reports the present status of this archive system, and the view of a subject and future

1. はじめに

J-PARC (Japan Proton Accelerator Research Complex) においては多数の機器により制御されており、Linac、RCS に関して約 64000 点もの EPICS レコードのデータを収集している。現状ではリレーショナルデータベースの PostgreSQL を利用したシステム^[1]にてデータアーカイブを行なっているが、増大し続けるデータに対してシームレスなシステム拡張が困難であること、ハードウェア老朽化に伴う換装時のデータ移行が大変であること等の課題が存在する。これらの課題に対応するため、分散処理フレームワークの Hadoop^[2]と分散データベースの HBase^[3]を利用した次世代アーカイブシステムの検討を行い、テストシステムを構築し稼働させている。

Hadoop・HBase の運用上の問題点としてマスターノードが単一障害点となる事が挙げられるが、2 台のサーバを Heartbeat^[4]と Pacemaker^[4]を用いた HA (High Availability) クラスタ構成にしてシステム可用性を向上させている。また、メタデータ保護のため、2 台のサーバ間で DRBD (Distributed Replicated Block Device)を用いたデータレプリケーションを実施している。9 台のスレーブノードを用いて合計約 50TB の HDFS (Hadoop Distributed File System)を構築し、この上で HBase を稼働させている。現在、試験的に Linac における約 6500 点の 1~60 秒周期のポーリング型データと約 13000 点のイベント型データのアーカイブを実施しており、安定稼働している。また、データ検索に関しては、現行システムと比較して応

答時間が 1/5 程度にまで短縮されているケースもある。

本稿では、このアーカイブシステムの現状や課題並びに今後の展望について報告する。

2. システム構成

本システムはクラスタ管理を行うマスターノードとして 2 台のサーバを Active-Standby の HA 構成にし、9 台のサーバをデータブロックの格納や処理を行うスレーブノードに割り当て、これらを Gigabit Ethernet で加速器制御系ネットワークに接続している。システム構成の概略を Figure 1 に、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。

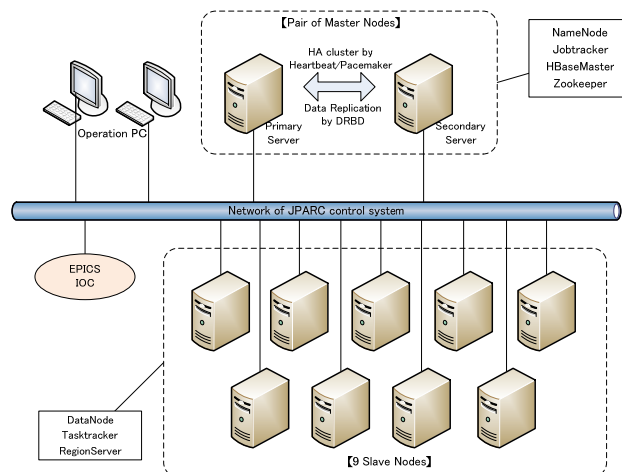


Figure 1: System structure.

[#] kikuzawa.nobuhiro@jaea.go.jp

Table 1: System Item Information

Role	Composition
Master Node (Primary)	DELL PowerEdge R610 CPU: Intel Xeon E5620 (4Core 2.4GHz) MEM: 24GB (8GB x3,1333MHz,DDR3) HDD: 600GB SAS 10krpm x4 (RAID10)
Master Node (Secondary)	DELL PowerEdge R200 CPU: Intel Xeon X3210(4Core 2.13GHz) MEM: 8GB (2GB x4,DDR2) HDD: 160GB SATA 7200rpm x1
Slave Node	DELL PowerEdge R410 CPU: Intel Xeon E5620 (4Core 2.4GHz) MEM: 24GB (8GB x3,1333MHz,DDR3) HDD: 2TB SAS 7200rpm x4 (RAID5)
Software	OS: CentOS6.3 (Japanese) Java: JDK 1.6.0_45 Hadoop 1.0.4 HBase 0.94.5 Heartbeat 3.0.7 Pacemaker 1.0.12 DRBD 8.4.1 Ganglia 3.4.0

各スレーブノードは 2TB の HDD を 4 基搭載しており、RAID5 構成で実効容量約 6TB である。ここから OS 等のシステム領域を 50GB 確保し、残りをデータ領域に割り当て、9 台のスレーブノードで約 50TB の HDFS を構築している。

3 可用性の向上とリソース監視

3.1 単一障害点対策

Hadoop・HBase について、スレーブノードは複数台構成で冗長性を有しており、単一のサーバ障害によるシステム停止やデータロスが防止される。一方、マスターノードは基本的にクラスタに 1 台しか無い単一障害点であり、サーバやサービスの停止がシステム停止に直結する。また、マスターノード上にはデータ管理に関する重要なメタデータが保存されているが、これが破壊されたり失われたりすると、スレーブノード上のデータに異常が無くてもデータへのアクセスが出来なくなり、実質的にデータロスと同じ状況となる。

これらの問題に対して、マスターノードを 2 台用いて、クラスタウェア Heartbeat と Pacemaker による Active-Standby の HA クラスタ構成とし、メタデータ保護の為にメタデータを格納するパーティションに対して、サーバ間データミラーリングツールの DRBD を用いたデータレプリケーションを実施している。

Heartbeat はノード上のサービス死活監視を行い、

Pacemaker はサービスの起動制御や異常が検知された際のフェイルオーバーを行う。以下に制御対象のサービスを列挙する。

- マスターノードの仮想 IP アドレス
- DRBD のレプリカマスター
- Hadoop 系サービス(NameNode、JobTracker)
- HBase 系サービス(HBaseMaster、Zookeeper)

1 分周期で各種サービスの死活監視を行い、サービス応答がタイムアウトした場合には従系サーバへとフェイルオーバーする。また、Heartbeat 自体がダウン若しくは制御不能な状態に陥ったと判断される場合には、STONITH (Shoot The Other Node In The Head)の機能を用いて強制的にノードをシャットダウンさせてフェイルオーバーさせる様になっている。

なお、HBase 系サービスの起動は Hadoop 系サービス起動時の初期化処理が完了してから行う必要があるため、Hadoop 系サービスの起動から 180 秒のタイムラグを設けている。フェイルオーバーは各種サービス切替・起動とこのタイムラグ等により、5 分程度の時間を要する。

3.2 リソース監視

本システムのように多数のサーバを用いたシステムの場合、サーバ台数が増加するに従い、各サーバのリソース使用状況確認が煩雑になってくる。

そこで、本システムでは Ganglia^[5]を用いて構成ノードの CPU やメモリ利用率、ネットワーク帯域等の各種リソースを一元的に監視出来るようにしている。図 2 の様に Web ブラウザで各ノードやシステム全体としての時系列的な負荷状況がグラフ表示されるため、高負荷状態の傾向を把握し、必要に応じてノードの追加やリソース増強等を判断するのに役立てられる。

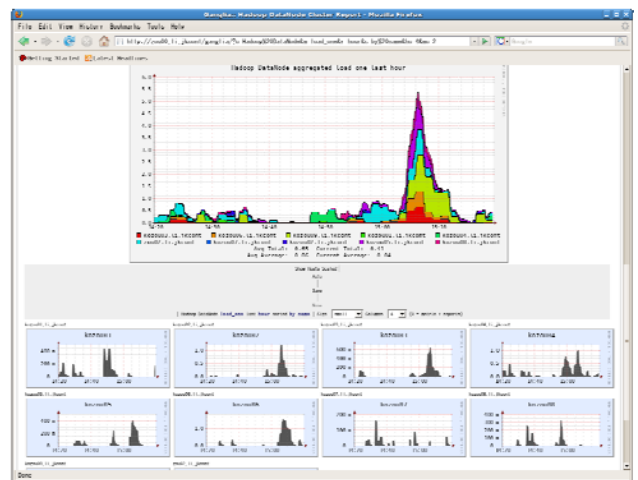


Figure 2: Resource monitoring by Ganglia.

4. データ構造と検索性能

4.1 データ構造の検討

HBase でのデータ構造については、これまでに大きく以下のようなパターンで検討・検証を行ってきた^[6]。

●現行踏襲型

現行システムに近い構造で、EPICS レコードを DTQ や RF 等機能別にグルーピングして 1 テーブルに格納する。row (一般的なデータベースシステムの「主キー」に該当)は値取得時間のタイムスタンプで、EPICS レコード名を付けた qualifier (一般的なデータベースシステムの「属性」に該当) 毎の column に格納する。

●Key-Value 型

EPICS レコード毎にその EPICS レコード名を含むテーブルを作成し、row は値取得時間のタイムスタンプで、取得値は row と 1:1 で紐付けする。

現行踏襲型は 1 つの row にグルーピングされた多数の取得値を紐付けているため、1 度の put 処理で多くの取得値をデータベースに登録出来る。そのため、データ登録時のシステム負荷が低いものの、検索時には qualifier に対する EPICS レコード名の検索処理が必要であるため、検索速度が遅いというデメリットがある。一方、Key-Value 型は row に対する検索のみで良いため検索速度が早いものの、データ書き込み時のテーブル接続セッションが多くなるため、データ書き込みクライアントとサーバ共に大きな負担が掛かる。

このテーブル接続の負担を抑えるために、Key-Value 型に於いて EPICS レコード毎にテーブルを持たせず、row を EPICS レコードと取得時間を繋げたものとし、その row と取得値を 1:1 で紐付けしたものを 1 つのテーブルに格納する構造とした。このデータ構造の概要を Figure 3 に示す。

HBase では従来のリレーショナルデータベースシステムでは困難であった巨大なサイズのテーブルを持つ事が可能であるため、1 テーブルに長期間且つ多種の EPICS レコードのデータを格納する事が可能となった。なお、テーブルは Linac、RCS 等の施設毎、周期取得のポーリング型と値に変化が発生した際に取得するイベント型の取得形式毎で分けている。

このデータ構造では、EPICS レコード毎にデータが時系列に並んだ状態であるため検索効率が良く、テーブル数が少ないためデータ書込時のテーブル接続のセッション数を抑える事が出来る。また、HBase はデータを row の昇順で格納しており、タイムスタンプの様に単調増加するものを row に割り当てた場合、特定のノードに書込処理が集中するという性質がある^[7]が、row の接頭辞に EPICS レコード名を付けることで書込位置が分散、つまりは書込処理対象の Region を保持するノードが自ずと分散化され、書込時の処理効率の向上が期待できる。

現在、Linac における EPICS レコードで約 6500 点の 1~60 秒周期のポーリング型データと約 13000 点のイベント型データを試験的に収集しており、安定的にデータアーカイブが行えている。

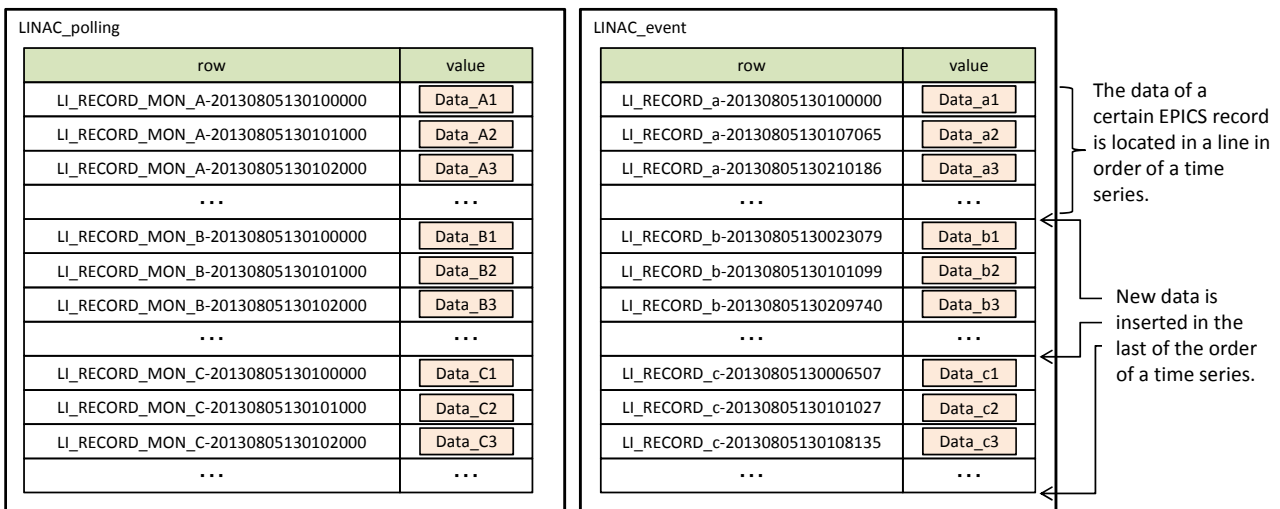


Figure 3: Image of data table structure.

4.2 データ検索性能

データ検索について、4.1 項で述べたデータ構造で HBase に現行データを移行した場合と、現行システムで性能の測定を実施した。現行システムサーバの主要スペックを Table 2 に示す。現行システムは非常に高速な StorNext^[8]をストア領域のディスクマウントに使用した書込用サーバと低速だが安価な外付け USB ディスクをストア領域に用いた過去データの検索用サーバを使用している。

Table 2: Spec of Present System

Role	Composition
for Data input	DELL PowerEdge R200 CPU: Intel Xeon X3210 (4Core 2.13GHz) MEM: 8GB (2GB x4, DDR2) Internal HDD: 160GB SATA HDD for Storing: 10TB (External file server storage is shared using "StorNext" in 10Gb network.)
for Data retrieving	DELL PowerEdge 860 CPU: Intel Xeon 3060 (2Core 2.4GHz) MEM: 4GB (2GB x2, DDR2) Internal HDD: 250GB SATA HDD for Storing: 4TB(1TBx4, Using external disk of USB connection)

HBase に移行した過去のアーカイブデータは Linac の 1 年分で、容量は約 1TB である。また、データは圧縮率と処理負荷のバランスに優れるとされる SNAPPY 形式^[9]の圧縮を行って HBase に格納している。

サンプルとして、LLRF 系と IS (Ion Source)系の EPICS レコードに対する測定値を 5 日分検索するのに要した時間を Table 3 に示す。なお、LLRF 系は現行システムで 1846 個の EPICS レコードのデータが紐付けられるグループで、IS 系は 45 個の EPICS レコードが紐付けられる。

検索で取得したデータ数は 1 秒周期のサンプリングで 5 日分約 43 万件であり、検索はキャッシュの無い状態で行ったものである。

Table 3: Data Retrieval Performance Comparison

System and Data Type	Time(sec)
Present System for Writing (LLRF Data)	50.5
Present System for Retrieving (LLRF Data)	100.0
HBase (LLRF Data)	20.9
Present System for Writing (IS Data)	16.8
Present System for Retrieving (IS Data)	26.9
HBase (IS Data)	20.9

現行システムでは、データベースの 1 レコードにグルーピングされた各 EPICS レコードの Double 型 8byte の取得値が byte 列で結合された状態で格納^[6]されており、検索には以下の様な手順が必要である。

- ① EPICS レコードの定義テーブルから検索対象のテーブル名称とデータ配列順序を取得する
- ② 検索対象のテーブル名を含むテーブルで検索期間に合致する日付のテーブルからデータを取得する
- ③ 取得した各データを 8 バイト毎に分離する
- ④ 分離したデータから①で取得した配列番号のデータを抜き出す

一方、HBase のシステムでは上記③④に該当するデータの分離・抜き出し処理が不要であるため、以下の手順のみで検索が行える。

- ① EPICS レコードの定義テーブルから、検索対象のテーブル名称を取得する
- ② 検索対象のテーブルの row に対して「EPICS レコード名+検索期間」でマッチするデータを取得

HBase のシステムでは、PostgreSQL による現行システムと比較して、書込用サーバに対する IS 系データの検索以外で応答時間が短縮されている。特に、現行検索用サーバに対する LLRF 系データの検索では 1/5 程度に時間短縮されている。現行書込用サーバに対する IS 系データの検索が HBase のシステムよりも早いのは、IS 系ではグルーピングされる EPICS レコードの数が 45 個と少なく、データ分離・抜き出し処理の所要時間が短いため、StorNext に依る高速なディスクアクセスの効果が大きい為である。ただし、大部分の EPICS レコードは数百～千数百個程度の多数でグルーピングされているため、HBase システムでは大部分の EPICS レコードに対して検索に要する時間短縮が見込まれる。

なお、純粋なディスクアクセス速度では StorNext を使用した現行書込用システムの方が優れているが、ディスクストレージや StorNext のソフトウェアライセンスのコストが非常に大きい。一方で HBase システムは比較的安価な汎用サーバを利用することで、現行システムよりもハードウェアやソフトウェアのコストを抑えながら大容量データアーカイブ環境を構築する事が出来る。また、必要に応じてスレーブノードを増設してディスク容量や性能の増強が行えるため、ミニマムスタートでシステムを導入出来、高いコストパフォーマンスを得られる事が大きな利点である。

5. 課題と今後の対応

テストシステムによるこれまでの試験利用と本格運用に向けての検討を通じて幾つかの課題等が浮上している。ここではその課題や対応の方向性について記述する。

5.1 データバックアップ

本システムでは、HDFS の機能であるスレーブノードのデータレプリカ保持や DRBD に依るマスターノードのメタデータレプリケーションにより、障害に対するデータロストの可能性を大幅に低減させている。ただし、不適切な操作によるデータ消失や災害等によるノードの多数損壊等によるデータロストも想定される。これらのリスクに対しては、スナップショット的なバックアップの取得や遠隔地へのレプリケーションサイトの設置等の方策が考えられるが、データ保護強度に応じて多額のコストを要するため、どのレベルまで対策を講じるかは今後の検討課題である。

5.2 システム可用性の向上

システム全体の可用性は単一障害点のマスターノードを HA クラスタ構成とすることで大幅に向上させているが、マスターノードのフェイルオーバーに 5 分程度の時間を要するため、データ収集クライアントの接続セッションがタイムアウトして以降のデータ収集が停止する恐れがある。そのために、収集プロセスの稼働状況を監視し、プロセスダウン時には再立ち上げを行う等の仕組みを実装する必要がある。

5.3 アプリケーションの移植

本番運用に向けて、現在使用している加速器運転データの表示や検索等を行う各種アプリケーションについては HBase システムに対応したものへと移植する必要がある。HBase との連携については基本的に Java の API を使用するため、Java 以外の言語で作成されたアプリケーションについてはソースコード再利用における制約が多く、移植の負担が大きくなることが懸念される。

これらのアプリケーションについては Thrift、REST、Avro の様なゲートウェイ API を通じたアクセスや、JNI (Java Native Interface) の利用可能性と共に、Java へのコードマイグレーションも視野に入れて検討する。

6. まとめ

HBase を利用した運転データアーカイブのテストシステムを構築して Linac の運転データのアーカイブや現行システム上のデータを移行して検索性能評価を行った結果、データ収集や検索に於いて十分に利用出来る性能や安定性を有することが確認された。

また、単一障害点であるマスターノードを HA クラスタ構成とすることでシステム可用性を向上させ、メタデータ領域のサーバ間レプリケーションによりメタデータ消失のリスクを低減させている。

今後は RCS や MR のデータ収集と過去のデータ移行を進めて加速器運転データアーカイブシステムとしての本格運用を目指す予定である。また、対応するアプリケーションの移行や開発を行い、将来のシステム利用に向けた準備を行うものとする。

参考文献

- [1] S.Fukuta, et al., "Development Status of Database for J-PARC RCS Control System (1)", Proceedings of the 4th Annual Meeting of Particle Accelerator society of Japan, August 2007
- [2] <http://hadoop.apache.org/>
- [3] <http://hbase.apache.org/>
- [4] <http://www.linux-ha.org/wiki/Heartbeat>
- [5] <http://ganglia.sourceforge.net/>
- [6] A.Yoshii et al., "J-PARC operation data archiving using Hadoop and HBase" Proceedings of the 9th Annual Meeting of Particle Accelerator Society of Japan
- [7] Apache HBase Reference Guide (<http://hbase.apache.org/book.html>)
- [8] <http://cn.teldevice.co.jp/product/detail/stornextt>
- [9] <http://code.google.com/p/snappy/>