EXAMINATION OF APPLYING HADOOP FOR J-PARC DRIVING DATA ARCHIVE

Akinobu Yoshii [A)], Nobuhiro Kikuzawa[A)]
[A)] Japan Atomic Energy Agency
2-4 Shirane Shirakata, Tokai-mura, Naka-gun, Ibaraki 319-1195

Abstract

J-PARC (Japan Proton Accelerator Research Complex) is controlled with a lot of equipment, and EPICS record of Linac and RCS extends to about as many as 64000 points. About 4.5TB driving data has been stored in the data archive system since 2006. It will be expected that driving data volume will increase by an enhancing of equipment and the more shortening at sampling intervals and so on. Finally total driving data volume will reach 100TB for the operation period. In the future, the data archive system will be needed that can correspond to high-speed writing and mass storage. However, it costs very much to introduce a large-scale data archive system at a time, the system with a flexible scalability is preferable. It is thought that Hadoop is powerful as the system base that meets such a requirement, and this report shows the examination of applying it.

J-PARC 運転データアーカイブにおける Hadoop 適用の検討

1. はじめに

J-PARC においては多数の機器により制御されており、現状の EPICS レコードは Linac、RCS に関するもので約 64000 点にも及ぶ。運用中はこれらの機器で取得される大量のデータを、データの同期性に応じて 2 種類のデータベースに収集している[1]。しかし、データ量増加に伴い、システムの書込み性能や大容量データ処理が頭打ちとなり、将来的な加速器設備の拡張やサンプリング間隔短縮等に対して柔軟に対応できなくなることが懸念される。

そのため、書込み性能に優れ、大容量データ保存に対応できるデータアーカイブシステムが後に必要になると考える。ただし、初期導入で大規模システムを一度に構築することは多大なコストが発生し、リソース利用効率も低下するため、スモールスタートで開始し、必要に応じて性能やストレージ容量の拡張が柔軟に行えるシステムであることが望ましい。

これらの要件を満たすシステムの候補として、分散処理フレームワーク Hadoop[2]の利用が有力であると考える。本稿では、この技術の運転データアーカイブシステムへの適用に関する検討内容について記述する。

2. Hadoop について

Hadoop とは Google 社で独自開発された多数の計算機による大規模クラスタの分散処理システム[3,4]をベースにオープンソース化されたフレームワークであり、複数のプロダクトから構成される。その中でも本目的に於いて特に重要なプロダクトとして分散ファイルシステム HDFS（Hadoop Distributed File System）およびこのファイルシステム上で稼働する分散データベース HBase[5]を挙げる。これらについて、それぞれの主な特徴を表 1、表 2 に示す。

表 1：HDFS の主な特徴

データ管理	ファイルはブロックに分割され、クラスタ内の DetaNode に分散配置される。配置情報は NameNode 上のメタデータで管理される。
ファイルアクセス	基本的に OS から直接ファイルシステムをマウントできず、コマンドライン及び API を経由する必要がある。
大容量ファイル対応	テラバイト、ペタバイトクラスの巨大なファイルを扱うことが可能。
可用性	複数台の DataNode でデータブロックのレプリカを保持し冗長性を確保している。ただし、Namenode が単一障害点となる。
拡張性	DataNode の追加で容量と性能の拡張が容易に行える。

表 2：HBase の主な特徴

DB 種別	列指向型（Key Value 型）の NoSQL DB に分類される。データ挿入と参照が中心であり、基本的に変更や削除の概念は無い。
データ形式	データは登録時にキーでソートされた状態で配置される。データ型は全て byte 形式となる。
データ容量	テラバイト、ペタバイトクラスの容量にも対応可能。
可用性	HDFS 同様に SlaveNode は冗長性が保たれるが、MasterNode が単一障害点となる。
機能制限	テーブル結合やクエリによる演算、ロールバック、トランザクション制御等の機能が無い。

3. 運転データ

J-PARC の運転データは定期的にデータが生成されるポーリング型とステータス変化が生じた際にデータが生成されるイベント型の 2 種類が存在する。いずれのデータ型も基本的にタイムスタンプと測定値から成るシンプルな構成である。なお、ポーリング型データについてはサンプリング間隔が 1～60 秒となり、ピーク時には秒間 1 万件程度のデータが発生する。

2006 年の J-PARC 運転開始以降、運転データの総容量は約 4.5TB となるが、今後の設備拡張やサンプリング間隔の短縮等に伴いデータ容量が加速度的に増加し、多く見積もった場合には最終的に 100TB にまで達することも予想される。これまでの総データ容量の推移と今後の予想を図 1 に示す。

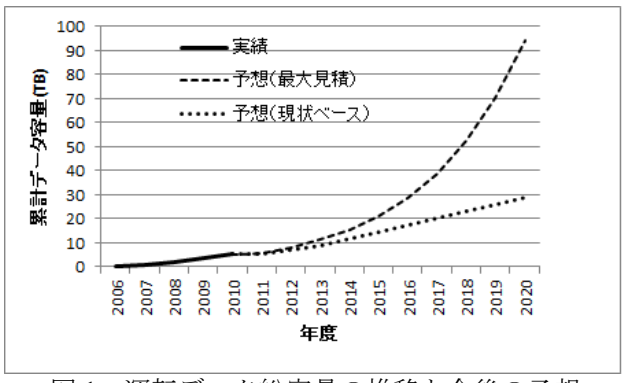

図 1：運転データ総容量の推移と今後の予想

4. 検証・評価

4.1 検証環境のシステム構成

今回検証・評価を行ったシステム構成を図 2 に、各計算機並びにソフトウェアの構成を表 3 に示す。なお、本来ならば、MasterNode は Master 系サービスのみを稼働させるのが適切であるが、計算機台数が少ないため、MasterNode にも SlaveNode としての機能を実装した。

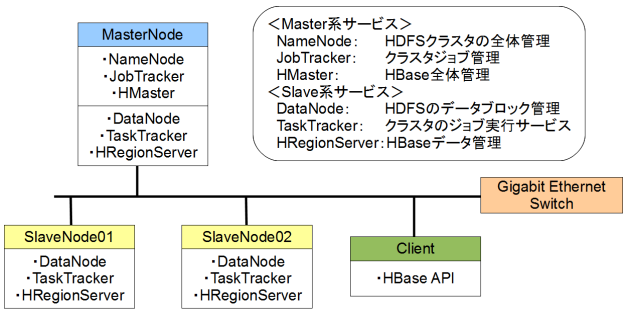

図 2：検証システム構成

表 3：検証器 HW スペック一覧

役割	主要スペック
Master Node	Lenovo Think Center M90z model 5205 CPU：Intel Core i3 530 (2Core 2.93GHz) メモリ：4.0GB (PC3-10600 DDR3) HDD：160GB SATA 7200rpm
Slave Node01	EPSON Endeavor MR4000 CPU：Intel Core i5 661 (2Core 3.33GHz) メモリ：2.0GB (PC3-10600 DDR3) HDD：500GB SATA 7200rpm
Slave Node02	lenovo Thinkpad T500 CPU：Intel Core2Duo T9600 (2Core 2.80GHz) メモリ：4.0GB (PC3-8500 DDR3) HDD：200GB SATA 7200rpm
Client	lenovo Thinkpad T520 CPU：Intel Core i5 2520M (2Core 2.5GHz) メモリ：4.0GB (PC3-10600 DDR3) HDD：320GB SATA 7200rpm
Software	OS：CentOS5.6 (日本語版) (※Client のみ Windows7＋Cygwin) java：JDK 10.6.2-1 (1.6.0-24) Hadoop：0.20.203.0 HBase：0.90.3

4.2 性能について

HBase の JAVA API を使用したデータの書込みとファイルへの出力を行うプログラムを作成し、Client 上でプログラムを実行した。書込みは 1 レコード当たり 10byte のデータを 10 カラム分登録するものとした。

検証は全てのクラスタ機能を MasterNode 上で実行する疑似分散モード、完全分散モードとして SlaveNode を 1 台、2 台追加した構成でそれぞれ同じプログラムを実行し、その実行時間から平均スループットを計測した。書込み時の結果を図 3、ファイル出力時の結果を図 4 に示す。

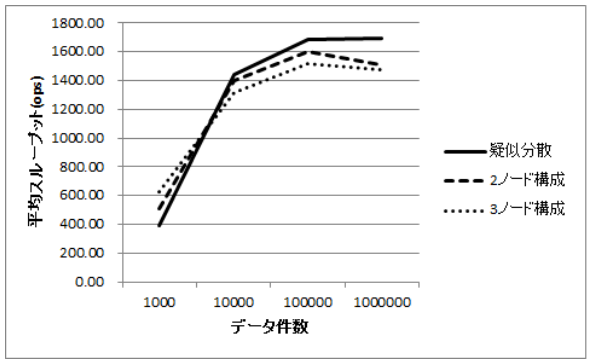

図 3：書込みスループット比較（構成別）

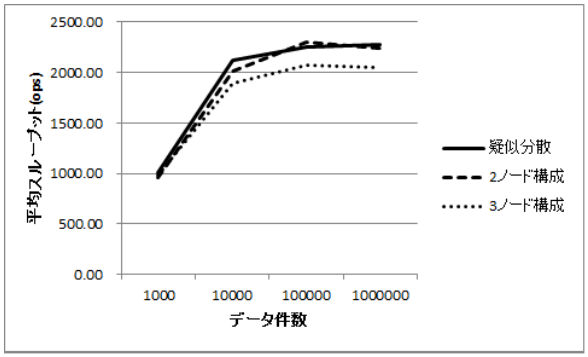

図 4：ファイル出力スループット比較（構成別）

いずれの構成でもデータ件数が 1000 件と少ない場合にはテーブルオープン・クローズ処理等の基本処理部の処理負荷割合が高くなり、スループットは低くなっている。また、検証結果からはノード数増加に伴うスループット向上が見られないが、これについては以下の様な原因が考えられる。

- 検証規模が小さいため、分散化やネットワーク処理のオーバヘッドが勝ってしまった。
- SlaveNode は性能面（特にメモリ容量やメモリクロック）で MasterNode に劣っており、ボトルネックとなってしまった。

なお、Hadoop・HBase は、大規模システム向けであるため、性能のスケールメリットを得るにはある程度の台数規模が必要とある[6]。今回の検証規模ではノード増設による性能スケールを得るには十分ではなかったと考えられる。

因みに、プログラム実行中の各ノードの CPU 利用率は 30～50%程度、ネットワーク帯域は 30%程度で推移していた。一方、メモリ利用率は 90%以上でスワップも頻発していたことから、HW による性能向上のためには、メモリやディスクの I/O 性能を向上させることが効果的と考える。

また、同時に多数のデータ書込ジョブが発生するケースを想定して、マルチスレッド化した書込みプログラムを 3 ノード構成下で 5 スレッド同時実行させた場合について計測した結果を図 5 に示す。

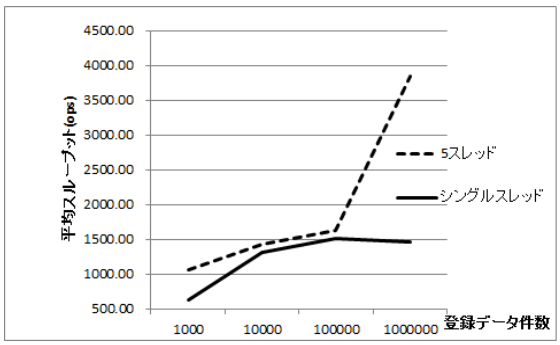

図 5：書込みスループット比較
（3 ノード構成スレッド数別）

マルチスレッド化に伴うリソース利用効率の向上により、スループットが向上している。ただし、今回の検証では各ノードの HW 性能に差があるため、データ登録処理にどのノードが割り当てられるかによって、スレッド毎の処理速度にかなりのバラつきが見られた。

なお、HBase はデータ登録時にキーの昇順でソートして格納するため、連番やタイムスタンプの様な単調増加する値をキーとした場合には特定のノードに処理が集中し、ボトルネックになる傾向がある[7]。今回の書込み試験で、キーを連番とランダム文字列とした場合の 2 パターンで実行したが、特段有意な性能差は見られなかった。これは検証規模が小さく、分散の偏りが起こりにくかったためと思われる。ただし大規模構成の場合には分散処理のメリットが十分に得られないことになるため、タイムスタンプをキーに用いる必要性がある場合には、適切なプレフィックスを付与するなど、登録処理が分散化されるような設計が必要と考える。

今回の検証では秒間 1500 レコード程度の書込み性能が得られており、秒間 15000 件程度のデータの登録に対応可能である。現状ではピーク時に秒間 10000 件程度のデータが発生するため、現状と同程度のデータ量であれば問題無く処理が可能と考える。ただし、将来的に処理データ量が増大した場合を考慮すると十分とは言い難い。この点については、HW 性能の向上や構成規模の拡大に加え、設定のチューニングやデータ登録プログラムにおける分散処理部の改善等、性能向上の余地は数多く残っている。スモールスタートの構成で効率良く処理するためにはハード・ソフトの両面で工夫が必要であるが、ポテンシャルは非常に高いと考える。

4.3 大容量データの扱いについて

大容量データ保管時の挙動を調べるため、HBase 上に膨大なレコード数を持つテーブルを作成し、データ検索を実行した。登録データは 1 つのテーブルに 4.2 節と同じデータ構成で 2 億件のレコード持つ総データ容量約 25GB のものとした。（データブロックのレプリカを持つため、ファイルシステムのディスク使用量はその倍以上となる）

データの登録には約 12 時間を要したが、データ登録自体は問題なく実行完了した。また、キーによる範囲検索に於いても、同じデータ数を抽出する場合では 100 万レコードを格納したテーブルに対して検索を行った場合と大差無い実行時間であり、大容量化による顕著な性能劣化は見られなかった。

なお、現状のシステムでは、装置と日付毎にテーブルを分割してデータを保管しており、大量のテーブルが DB 内に存在している。HBase を用いることにより少数の大容量テーブルでデータを一元的に保管することができれば、データの管理や利用が簡便となることが期待される。

4.4 拡張性について

Hadoop・HBase の大きな強みとして柔軟な拡張性が挙げられるが、実際に稼働中のクラスタシステムに新しい SlaveNode を追加してその挙動について確認を行った。

追加時のオペレーションとしては、以下の通りで非常に簡便であった。

- 追加 SlaveNode の構築（Hadoop・HBase の設定は既存 SlaveNode からのコピーで良い）
- MasterNode の設定ファイルに増設 SlaveNode のエントリを追記
- サービスの再起動

追加操作後はクラスタシステムへのノードエントリの追加処理とファイルシステムの容量増加が速やかに行われ、その後暫くして自動的に追加ノードに対してデータの再配置が行われることを確認した。なお、サービス再起動時に 1～2 分程度のサービス停止が発生するが、一般的なシステム拡張のケースと比較すると停止時間が極めて短く、非常に拡張性に優れたシステムと言える。

4.5 可用性について

J-PARC は一旦運転を開始すると数週間昼夜を問わず連続で稼働する。データアーカイブシステムの停止は貴重な実験データの取りこぼしに繋がるため、運転中のシステム可用性確保は極めて重要である。今回は Hadoop・HBase の可用性の評価として稼働中に一部のノードを意図的に停止させ、サービスの継続性とデータ保護性能について確認を行った。

SlaveNode を停止した際には、MasterNode より障害検知がされるまでは、当該ノードで保持するデータブロックへのアクセス待ちが若干発生したが、障害検知後はクラスタ内で保持するレプリカデータにより自動的にデータへのアクセスが再開し、更に別の正常ノード上に不足するレプリカを再生成する挙動を示すことを確認した。障害ノードの復旧時には、障害ノード上のサービス再開のみでシステム全体のサービス稼働への影響は特段見られず、SlaveNode の可用性は非常に高いと思われる。

また、この性質を応用すれば HW 換装時において不可避であるデータ移行の作業負担やサービス停止の影響を大幅に軽減できる可能性があることが期待される。

一方、MasterNode は単一障害点となるため、ノードの障害がクラスタシステムの停止に直結する。また、データ保存場所を管理するメタデータが破損してしまった場合には、DataNode 上のブロックデータが正常であってもデータへのアクセスが失われてしまうため、メタデータの保護は極めて重要な課題となる。これらに対しては、MasterNode の HA（High Availability）クラスタ化とメタデータのレプリケーションを実施するような対応が必要であると考える。

5. まとめ

今回は Hadoop・HBase に関して加速器運転データのアーカイビングに対する適用可能性を探るという観点から検証を実施した。

柔軟な拡張性と大容量データ対応では今後も増大し続ける加速器運転データのアーカイブには最適であると考えられるが、実際の適用に向けては性能の向上と MasterNode の可用性確保が重要な課題となる。また、一般的なデータベースと比較すると分散データベースは歴史も浅く、システム構築やプログラミングに関する事例や情報がまだまだ少ないため、検証を重ねてこれらのノウハウを蓄積する必要がある。

参考文献

[1] H.Takahashi et al., “Development Status of Database for J-PARC RCS Control System”, Proceedings of the 4th Annual Meeting of Particle Accelerator Society of Japan and the 32nd Linear Accelerator Meeting in Japan

[2] http://hadoop.apache.org/

[3] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, The Google File System, pp. 29-43, Cécile Roisin, Ethan V. Munson, Christine Vanoirbeek (Ed.), 19th ACM Symposium on Operating Systems Principles, Bolton Landing, New York, October 2003.

[4] Fay Chang, Jerey Dean, Sanjay Ghemawat,Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows,Tushar Chandra, Andrew Fikes, and Robert E.Gruber. Bigtable: A distributed storage system forstructured data. In In Proceedings of the 7thConference on USENIX Symposium on OperatingSystems Design and Implementation - Volume 7,pages 205-218, 2006.

[5] http://hbase.apache.org/

[6] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears, “Bencimarking Cloud Serving Systems with YCSB” Proceedings of the 1st ACM symposium on Cloud computing, June 2010.

[7] http://oss.infoscience.co.jp/hbase/book/book.html