J-PARC ハドロンビームライン用データアーカイブシステムの開発(2)

DEVELOPMENT OF DATA ARCHIVE SYSTEM FOR J-PARC HADRON BEAMLINE(2)

豊田晃久#, 上利恵三, 青木和也, 家入正治, 加藤洋二, 倉崎るり, 里嘉典, 澤田真也, 白壁義久, 高橋仁, 田中万博, 広瀬恵理奈, 皆川道文, 森野雄平, 山野井豊, 渡辺丈晃

Akihisa Toyoda #, Keizo Agari, Kazuya Aoki, Masaharu Ieiri, Yohji Kato, Ruri Kurasaki, Yoshinori Sato, Shinya Sawada, Yoshihisa Shirakabe, Hitoshi Takahashi, Kazuhiro Tanaka, Erina Hirose, Michifumi Minakawa, Yuhei Morino, Yutaka Yamanoi, Hiroaki Watanabe

KEK

Abstract

As the data archive system of the J-PARC hadron Beamline, we have been using the Channel Archiver since starting operation in 2009. However, since the Channel Archiver has not been maintained for nearly 10 years since 2006, it is necessary to shift to the next system. This time we will present performance improvement and performance comparison of RDB Channel Archiver, and future prospects. We will also introduce the network load reduction and monitoring system related to the above.

1. はじめに

J-PARC ハドロン実験施設では EPICS(Experimental Physics and Industrial Control System)[1]データアーカイブシステムとして、Channel Archiver[2]を運転開始以来使用してきた。しかしこの Channel Archiver は 10 年以上メンテナンスされていないため、次期アーカイブシステムへの移行が急務となっている。

我々はその候補の一つとして、前回[3]発表したように、CSS[4]に付属する RDB(Relational Data Base) Channel Archiver を使用している。今回は前回問題になった運転時の読み出しスピード不足の問題を解決するために、主に 2 つの対策を実施した。一つはバッファまわりの設定のチューニングで、もう一つはデータベースのリプリケーションの採用である。この対策後の性能試験を行ったので報告する。また、ネットワーク負荷対策やそのモニタリングについても触れる。

2. アーカイブシステム構成

Figure 1 にネットワークおよび監視システム、EPICS、アーカイブシステムの構成を示す。従来の Channel Archiver 側は前回から変わっておらず、書き込み専用の server1 と xmlrpc プロトコルを利用した読み出し専用の server2 の 2 台構成となっている。Channel Archiver 専用の EPICS CA(Channel Access)のゲートウェイ(以下 GW)[5]は server1 で走らせている。Figure 1 には明示していないが、ストレージは Channel Archiver 専用の NAS(Network Attached Storage)(2 TB, RAID6)を使用している。また GUI クライアントが使用する GW はアーカイバーとは別に専用で server3 を用意している。RDB Channel Archiver 側は書き込み専用の MySQL master である server5 とその GW である server4、読み出し専用の MySQL slave である server6 の 3 台構成となっている。ストレージは RDB Channel Archiver 専用の NAS(10 TB, RAID6 構成)を使用している。

アーカイブシステムとしては、前回運転中で書き込み負荷がある場合に読み出し性能が劣化するという問題があった。1 つ目の対策であるバッファまわりの設定変更としては、グローバルバッファおよびスレッドバッファ、および最大コネクション数を適切に再設定し、メモリーが適切に使用できるようにした。クエリーバッファーは使用しない設定とした。上記の結果、総メモリー使用量は平均約 5 GB と全体の 1/3 程度となった。リプリケーションについてはすでに紹介したが、master 1 台に対して slave 1 台を増設し、slave 側から読み出すようにして負荷分散を図った。

EPICS としては、データ量の多い EPICS channel が増えるにつれて EPICS channel への接続に時間がかかったり、更新が遅れたりして対処が必要な場合が出てきた。よって、今回から GW の構成を変更している。今までは 1 個の GW だけで他施設からのアクセスも含めてすべて賄っていたが、今回は 2 節に示したように 3 つの GW を用意し、かつそれぞれの接続先を制限した。よってそれぞれの GW ログを見ることにより、原因がどこにあるのかを判別できるようになった。

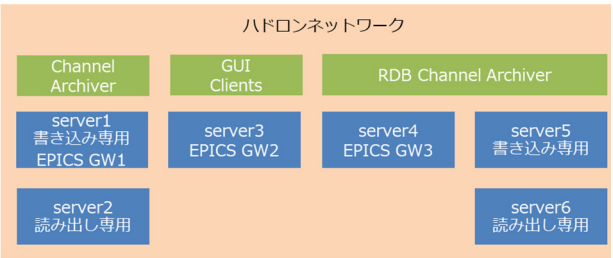

Figure 1: EPICS and Archive system framework.

上記に関連してネットワーク問題が生じるようになってきたので、監視システムを強化した。具体的には mrtg を導入し、全クライアントおよびサーバーの CPU 負荷、ネットワーク負荷、メモリー負荷、ディスク消費量を監視および過去の履歴を参照できるようにした。今後は標的温度など重要な EPICS IOC(Input Output Controller)も監視対象にする予定である。また RDB Channel Archiver の MySQL データベースについては munin を導入して、MySQL のスループットやスロークエリ数、接続数、バッ

akihisa.toyoda@j-parc.jp

ファ状態、slave 遅延時間などを監視できるようにした。munin サーバーについても munin で apache と munin の状態を監視できるようにした。運転中監視を継続し始めてからは、大きな問題が生じていないことを確認できている。その原因としては上記の GW の強化対策が効いているものと考えられる。

3. 試験内容とその結果

試験としては以下の内容を行った。
試験期間：2017 年 6 月から 7 月。一部の試験はビームタイム終了後に行っている。2017 年は 2016 年に比べて若干(1%程度)チャンネル数が増えている。
試験内容：

- データ取得速度の測定
- 書き込み速度の測定
- データ一致測定

3.1 データ取得速度の測定

この節ではデータ取得速度の測定について述べる。測定期間は 2017 年 6 月から 7 月の運転データを使用した。データ取得速度の測定方法は前回の報告[3]を踏襲した。具体的には以下のようになる。

- Scaler データとして、ビーム強度を採用。1 spill(5.52 秒)に 1 回更新(1 point/spill)で 1 ch(1 ch/point)である。低負荷データの代表とする。
- Array データとして、ビームプロファイルを採用。1 spill(5.52 秒)に 1 回更新(1 point/spill)で 64 ch(64 ch/point)である。高負荷データの代表とする。

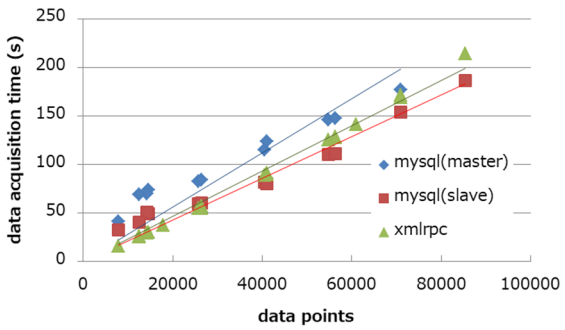

Figure 2: Fitting of data acquisition time.

前回は 0.1 秒更新のデータ(Fast Sampling)も解析したが、Channel Archiver 側で欠けることが分かっているので今回は採用しない。データ取得クライアントは前回と同じ python-xmlrpc および python-MySQL ライブラリを使用した。また前回の結果として、取得速度のボトルネックはネットワークではなくクライアントの性能に主に依存していることが分かっているため、最も性能の高いサーバーの一つである server5 でクライアントを実行している。

Figure 2 にデータ取得速度測定のためのフィッティングの例を示す。方法は前回と同様である。横軸 60000 points が約 4 日弱分になる。青ひし形が RDB Channel Archiver (master)の結果、赤四角が RDB Channel Archiver (slave)の結果、緑三角が Channel Archiver の結果を示している。それぞれ 1 次関数でフィットして傾きを評価した。

Table 1: Fitting Result (under Beam Operation)

データセット	Channel Archiver の DAQ レート	RDB Channel Archiver (master)の DAQ レート	RDB Channel Archiver (slave)の DAQ レート
Scaler データ (クライアントは server5)	1.4 x 10^3 points/s 1.4x10^3 ch/s	1.6 x 10^3 points/s 1.6 x 10^3 ch/s	3.5 x 10^3 points/s 3.5 x 10^3 ch/s
Array データ (クライアントは server5)	4.3 x 10^3 points/s 2.7x10^4 ch/s	3.6 x 10^3 points/s 2.3 x 10^4 ch/s	4.7 x 10^3 points/s 3.0 x 10^4 ch/s

Table 1 にビームタイム中の結果を示す。前回と比較すると、Scaler については Channel Archiver の性能はほぼ変わっていない。RDB Channel Archiver については、master の性能は 3 割ぐらい低下しているが、slave の性能は前回の master に比べて 1.5 倍程度向上している。通常 slave 側からデータ取得するので大幅な改善がなされていることになる。master 側で性能低下しているのは、データ収集だけでなく、間欠的に slave へのデータ提供もするようになったためと考えられる。

Array については Channel Archiver の性能はほぼ前回を再現している。RDB Channel Archiver の master では前回 Channel Archiver に比べて約 1/4 の性能しか出ない問題があったが、今回はほぼ同等の性能が出ている。これは MySQL のチューニングを行ったことによると考えられる。slave については Channel Archiver より若干優れた性能となっている。

Table 2: Fitting Result (after Beam Operation)

データセット	Channel Archiver の DAQ レート	RDB Channel Archiver (master)の DAQ レート	RDB Channel Archiver (slave)の DAQ レート
Scaler データ (クライアントは server5)	1.4 x 10^3 points/s 1.4x10^3 ch/s	2.3 x 10^3 points/s 2.3 x 10^3 ch/s	5.2 x 10^3 points/s 5.2 x 10^3 ch/s
Array データ (クライアントは server5)	4.3 x 10^3 points/s 2.7x10^4 ch/s	4.5 x 10^3 points/s 2.9 x 10^4 ch/s	5.6 x 10^3 points/s 3.5 x 10^4 ch/s
Large Array データ (クライアントは server4)	2.2 x 10^1 points/s 6.6x10^4 ch/s	2.3 x 10^3 points/s 7.0 x 10^4 ch/s	2.4 x 10^3 points/s 7.3 x 10^4 ch/s

続いて Table 2 にビームタイム後の結果を示す。Scaler については、Channel Archiver と RDB Channel Archiver の性能を比較すると、master で約 1.5 倍、slave で約 4 倍

の性能が出ている。Array については前回 Channel Archiver より RDB Channel Archiver の性能が若干劣っていたが今回は上回っている。特に slave では 3 割程度読み出し性能が上回っている。またビームタイム後については更にデータ点の多い Large Array データも解析した。対象は 2 次粒子数モニタで、1 spill(5.52 秒)に 1 回更新(1 point/spill)で 3000 ch/point のデータとなる。これについては Channel Archiver も RDB Channel Archiver も大きな性能差は出ていない。これはボトルネックとしてクライアントの性能が大半を占めるようになったためと考えられる。実際最初は server5 で解析したが、この場合はクライアント負荷がかかりすぎて MySQL master サーバーが遅くなってしまい正しい評価ができなかった。よって同等の性能の server4 を利用して解析した。1 秒あたりのチャンネル数で言えば 70000 ch/s あたりにクライアントの処理速度の限界があるものと考えられる。これ以上向上させるには server の性能向上を図るか、もしくはクライアントを高速なものに変えるかする必要がある。

3.2 書き込み速度の測定

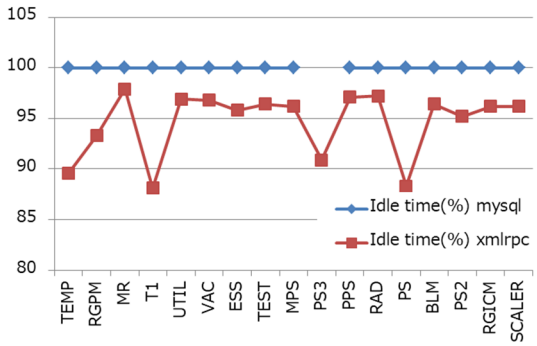

Figure 3: Process idle time.

この節では書き込み速度が十分かどうかを評価する。Figure 3 に各アーカイバーの idle 時間(%)を示す。PS3 プロセスは RDB Channel Archiver 側にはないので空欄になっている。負荷分散のため、Channel Archiver/RDB Channel Archiver ともに複数のプロセスに分かれていて、その名称が横軸になっている。前回同様 Channel Archiver 側で若干負荷が高いが、問題になるほどではない。

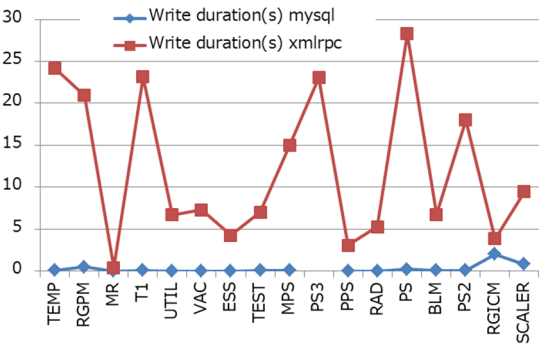

Figure 4: Write duration time.

Figure 4 には各アーカイバーの書き込み時間(s)を示す。どちらのアーカイバーも書き込み周期は 30 秒のため、30 秒以上かかると書き込みが間に合っていないことになる。プロセス間のチャンネル数の調整の結果、前回とは異なり Channel Archiver 側でも 30 秒以内に収まっていてデータ欠けが生じにくくなっていることが分かる。

つづいて Figure 5 に RDB Channel Archiver で Large Array データ解析中の書き込み時間(赤四角)と通常時(青ひし形)の書き込み時間の比較を示す。Large Array データ解析中はクライアントによる CPU 負荷およびメモリー消費が激しいため通常ほとんど書き込み時間がかからないのに対して最大で 7 秒程度かかるようになっている。それでも書き込みが間に合わなくなるほどではなかった。

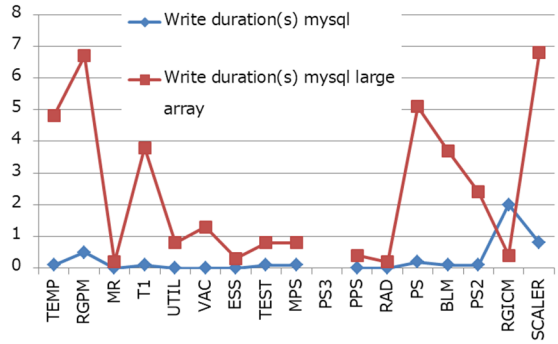

Figure 5: Write duration time under large array data acquisition.

3.3 データ一致測定

前回同様の方法で、データ一致測定を行った。具体的な方法を再掲すると、Channel Archiver と RDB Channel Archiver からデータを取得し一致するかどうかをチェックしている。よって両者ともに欠損した場合は検知できない。前回全データ解析はすでに行っているので、今回は上記 3.1 節のデータ解析中で読み出し負荷がかかっている状態のデータ一致確認のみを行った。その結果 Scaler データ取得中および、Array データ取得中の場合においてデータの欠損は見られなかった。また試しに Large Array データを取得してみた時についても調べてみたが、データの欠損は見られなかった。よってビームタイム中の読み出しについて、データ欠け問題は生じないと考えられる。

4. まとめ

前回に引き続きデータアーカイブシステムとして従来の Channel Archiver に加えて RDB Channel Archiver を運用し、性能評価を行った。前回問題になった運転中で書き込み負荷がある状態での RDB Channel Archiver の性能劣化については MySQL データベースのチューニングとリプリケーションの採用で遜色ない性能を出すことができた。また Channel Archiver および RDB Channel Archiver ともにデータ欠けなどの問題は生じておらず、安定して運用できるようになっている。またネットワーク安定性については EPICS CA Gateway の複数設置によって向上しており、またその安定度は複数の監視システム

によって常時監視できるようになった。今後は Archiver Appliance など別のアーカイバーの試験導入を検討しているところである。

参考文献

[1] EPICS Web page; http://www.aps.anl.gov/epics/
[2] Channel Archiver Web page;
https://ics-web.sns.ornl.gov/kasemir/archiver/
[3] A. Toyoda *et al.*, “DEVELOPMENT OF DATA ARCHIVE SYSTEM FOR J-PARC HADRON BEAMLINE”, Proceedings of the 13th Annual Meeting of Particle Accelerator Society of Japan (2016) MOP091, pp 630-633.
[4] Control System Studio Web page;
http://controlsystemstudio.org
[5] EPICS CA Gateway; https://github.com/epics-extensions/ca-gateway/releases