# SuperKEKB におけるデータアーカイブの読み出し高速化
# IMPROVEMENT OF DATA ARCHIVE READING SPEED FOR SUPERKEKB

森田昭夫 *
Akio Morita*
KEK

*Abstract*

In the SuperKEKB accelerator operation, the "KEKBLog" data archiving system, which is developed in KEK for the KEKB B-factory accelerator, records accelerator control parameters and monitor values. The KEKBLog archives are used for analyzing parameter history in trouble shooting and rendering semi real time trend graph to visualize operation status. In such KEKBLog usage, the data retrieving speed is very important. However, the data retrieving speed was not fast enough during KEKB B-Factory operation. The 20 times readout performance improvement is achieved by installing the cache storage server based on flash storages between the KEKBLog storage servers and the data retriever system. The flash cache storage system design and performance are reported in this article.

## 1. はじめに

SuperKEKB 加速器 [1, 2] は、Belle II 実験のための B 中間子生成を主目的とする非対称エネルギー (7×4GeV) 電子陽電子衝突型加速器であり、2016 年 2 月 1 日から 6 月 28 日にかけて Phase1 と呼ばれる試験運転が実施し、現在は物理実験へ向けた準備のために改装作業を行なっている。SuperKEKB 加速器の運転制御システムは、主に EPICS (Experimental Physics and Industrial Control System)[3] ツールキットを用いて構築されている。EPICS ツールキットによって抽象化された加速器の運転制御は、EPICS チャンネルアクセスプロトコルを通じて個々の機器を制御する IOC(Input/Output Controller) 上に存在する制御パラメータやモニター値を格納した実行時データベースに対する読み書きとして実行される。

加速器運転状態の履歴を残すために IOC 上の主要な制御パラメータやモニター値をチャンネルアクセスプロトコル経由で KEKBLog と呼ばれるアーカイビングシステムにて記録している。KEKBLog で記録されたアーカイブは、障害時の履歴調査や長時間の運転パラメータ相関分析等での利用以外に、数十分から数日のスケールで運転状態を可視化するためのトレンドグラフ作成にも利用されており、恒常的に多数の読み出しタスクが実行されている。アーカイビングシステムからの読み出し時間は、障害対応時の履歴調査の作業性や長時間に渡る相関解析の作業効率、トレンドグラフの応答速度に直結するため、読み出し時間短縮が求められている。

KEKBLog アーカイビングシステムのストレージ層と読み出し系の間に、近年安価に調達可能となった SSD を用いたキャッシュサーバーを導入することで、読み出し時間を大きく削減することに成功したので、これを報告する。

## 2. KEKBLOG アーカイビングシステム

KEKBLog[4] は、SuperKEKB 加速器の前身である KEKB B-Factory 加速器向けに KEK 内で開発されたデータアーカイビングシステムで、EPICS チャンネルアクセスプロトコル経由でデータを収集しストレージ上に格納するアーカイビングプロセス"kblog" とストレージ上に格納されたアーカイブからデータを抽出する"kblogrd" の二種類のプログラムで構成されている。KEKBLog のアーカイビングシステム及び読み出し系のシステム構成は、Figure 1 に示す通りである。KEKBLog のデータ記録層は、アーカイビングプロセス"kblog" が動作する収集サーバー上の最新データを記録したローカルストレージと、古いデータを蓄積する複数の NAS アプライアンスで構成されている。アーカイブからデータ抽出を行なう"kblogrd" は、収集サーバーと独立した運転制御用の CPU サーバー上で実行される。個々のサーバーや NAS アプライアンスは、10Gbps イーサネットを基幹としたバックボーンに 1Gbps イーサネットで接続されており、アーカイブされたデータの管理や抽出のためのファイルアクセスは、NFSv3 プロトコルによって行なわれている。

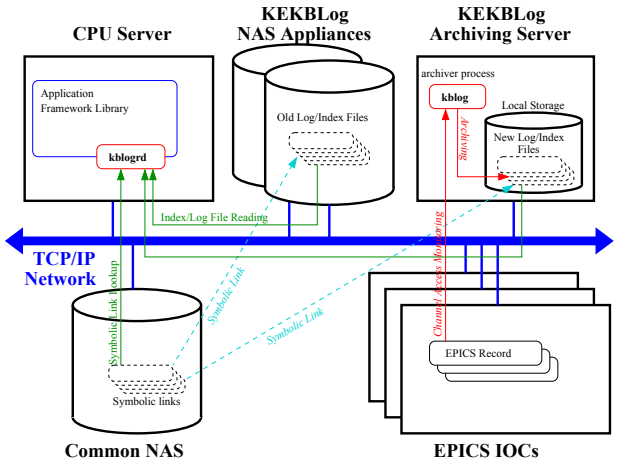


Figure 1: Block diagram of KEKBLog archiving system. Solid arrows show data flow. Dashed arrows show path reference by symbolic links.

* akio.morita@kek.jp

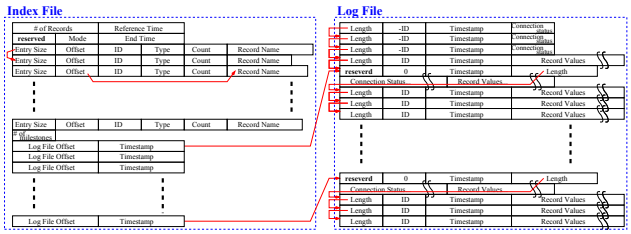


Figure 2: Log & index file format layout of KEKBLog archiving system. Red arrows show the reference pointer between contents.

収集対象となる EPICS レコードは、幾つかのグループを成し、グループ毎にアーカイビングプロセスが割り当てられている。アーカイビングプロセス”kblog” は、EPICS チャンネルアクセスプロトコル経由で受信した EPICS レコードの接続状態と値の変化毎に、イベント長、レコード識別子、タイムスタンプからなる固定長ヘッダーとレコード値等の可変長データをログファイルへ受信順に追記してゆく。また、途中からの読み出しを効率化するためにログファイルには、定期的に記録対象の全レコードの接続状態と値を纏めたマイルストーンが挿入される。ログファイルの記録と同時に、レコード識別子に対応する EPICS レコードの種類・長さ・名前のテーブルとログファイルへ挿入したマイルストーンのタイムスタンプとファイルオフセットの対応を記録したインデックスファイルが生成される。ログファイルとインデックスファイルの様式と参照関係の概要を Figure 2 に示す。インデックスファイル内のファイルオフセットには、符号付 32bit 整数が割り当てられているため、ログファイルは 2GiB もしくは一定記録時間ごとに分割されている。生成されたログ・インデックスファイルは、収集サーバーと NAS 上の複数のボリュームに分散して保管されており、ストレージの増強に伴い参照側への NFS マウントポイントの追加は随時行なわれる。ストレージ増強や収集サーバーから NAS へのファイル移動等に依存せずデータを読み出せるよう、全てのログ・インデックスファイルへのシンボリックリンクを共通系の NAS 内の決まったファルダーに集積し対応関係を維持している。

“kblogrd” による読み出しでは、共通系の NAS 上に集められたシンボリックリンクのリストと読み出し開始時刻から当該時刻のイベントを含むログ・インデックスファイル名が特定される。読み出し開始時刻を含むインデックスファイル中のマイルストーンテーブルから読み出し開始時刻直前のマイルストーンに対応するファイルオフセットを取り出し、ログファイルの頭出しを行いマイルストーンから読み出し対象レコードの初期値を得る。以降は、ログファイルに記録されているイベントを順次読み取り、対象レコードの変化を辿る操作を読み出し終了時刻まで繰り返す。ログファイル中のイベントは可変長かつ典型的なブロックストレージの割り当て単位よりも短く、特定のレコードに対応するイベントの分布に関する情報は記録されていないため、典型的な読み出し負荷はログファイルの連続読み出しとなり、ファイルキャッシュや先読み最適化等が効果的に働く一方で、読み出し対象区間に含まれるログファイルの全域を読み出すこととなる。

2010 年に運転を終了した KEKB B-Factory 加速器で運用した KEKBLog アーカイビングシステムでは、登録レコード数の多い収集グループからの読み出しが極めて遅く、季節変動を可視化するために数ヶ月から一年単位のデータを読み出すのに 30 分以上の時間 (数時間かかるケースも) を必要とし、利用者からは強く改善を望まれていた。当時は、KEKBLog からの読み出し速度は、収集サーバーや NAS からの I/O が律速となっていると信じられており、“kblogrd” を利用するアプリケーション向けのラッパーライブラリ側で読み出し済みデータの再利用を行なう等の対策は取られたが、本質的な改善は成されていなかった。

## 3. KBLOGRD 性能限界

“kblogrd” での読み出し性能が通説通り NFS 越しのファイルストレージからの読み出し速度で制限されているのであれば、近年のフラッシュメモリの容量単価下落に伴い比較的安価に大容量の構成が可能になった SSD を用いたローカルファイルシステムにアーカイブを複製し、“kblogrd” によるログファイルの読み出しを広帯域のローカルファイル I/O へ置き換えることで、大幅な読み出し性能の向上が期待できる。上記仮説の検証と、SSD 導入時の性能評価のため、読み出しが遅い収集グループの一つである RF/AnalogData から KEKB B-Factory 運転中の 2009 年のアーカイブ 1 年分 (約 120GiB) を複製して、ストレージ構成や”kblogrd” のコードに調整を加えながらアーカイブ読み出し速度のベンチマークを Table 1 に挙げるシステム上で行なった。

Table 1: Benchmark System Specification

| | |
|---|---|
| Processor | Phenom II x6 1090T |
| Memory | DDR3 16GiB(Dual Channel) |
| System Storage | 96GiB SSD |
| Cache Storage | SATA3 500GiB SSD x1 ∼ 3 |
| OS | FreeBSD[5]/amd64 9-STABLE |
| File System | ZFS[6] |

最初期のベンチマークでは、ログファイルの読み出し速度が “kblogrd” プロセス時に 100MiB/sec 以下で飽和した。これは、当時の標準的な 7200 回転のハードディスクの外周部での連続読み出し帯域未満であり、SSD の帯域を活用するには遅すぎる。この際の “kblogrd” プロセスの実行時間プロファイルの解析から、次のようなのボトルネックが発見された。

- memcpy(3) による同一データの多重コピー
- タイムスタンプ変換での浮動少数点演算
- イベント毎のレコード名の照合
- strftime(3)/localtime(3) 呼び出し毎の暗黙のメモリ割り当て・TZ 環境変数の解析

同一データの多重コピーは、 fread(3) による読み出しと一連の memcpy(3) を mmap(2) とポインタ渡しで置き換えることで、不要なメモリコピーを削減した。それ

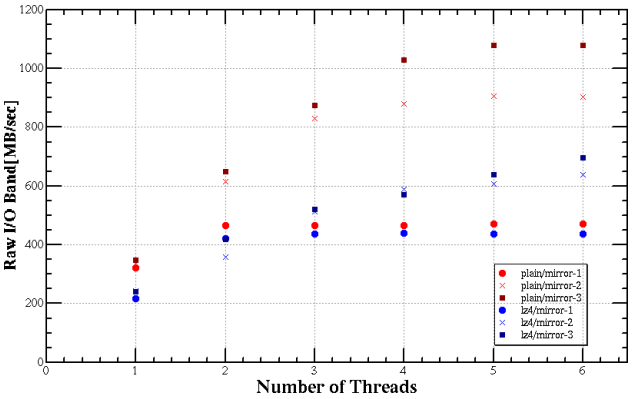


Figure 3: Raw storage I/O bandwidth of kblogrd on different storage configuration. The color of marks(red & blue) show plain(without compression) ZFS and LZ4 compressed ZFS, respectively. The circle, cross and box marks correspond with read access multiplicity of storage backend($1 \sim 3$).

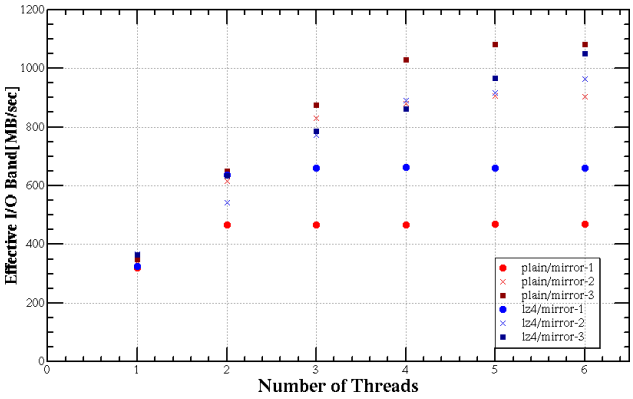


Figure 4: Effective I/O bandwidth of kblogrd on different storage configuration. The bandwidth for lz4 datas(blue marks) are scaled by compression ratio.

以外のボトルネックとなっていた処理は、演算強度の小さな同等の処理を新たに実装して置き換えた。一連のボトルネック解消後の”kblogrd” の処理速度は、同一システム上で 1 プロセスで 348MiB/sec の処理速度 (無圧縮時) を達成した。

改良された”kblogrd” を用いて、データを格納する ZFS の構成を SSD の並列数 (I/O 帯域) と透過的圧縮の有無 (無圧縮/LZ4 圧縮) を変えた際に、読み出しスレッド数に対して SSD の読み出し帯域がどのように変化するかを調べた結果を Figure 3 に示す。赤点で示される無圧縮 (plain) の ZFS 上では、“kblogrd” の I/O 帯域がスレッド数増加に従って増加し、SSD の並列数 (mirror-#) 毎に固有の上限で飽和している様子が確認できる。並列数 2 での上限が並列数 1 での上限のほぼ 2 倍であり、1 台あたり 470MiB/sec という帯域から、これは SSD の物理帯域の飽和と考えられる。また、3 並列での帯域が 1090MiB/sec で飽和しているのは、旧式化しているテストシステムのノースブリッジ (AMD 880G)- サウスブリッジ (SB850) 間の接続帯域もしくは SB850 の SATA コントローラの処理能力の飽和と考えられる。一方で、青点で示される LZ4 圧縮の ZFS 上では、並列度 1 のケース (mirror-1) を除いて、最大スレッド数 6 の時点で SSD の帯域は飽和していない。Figure 4 に示すのは、Fig.3 の I/O 帯域を LZ4 圧縮を加味して”kblogrd” の処理帯域に焼き直したものである。“kblogrd” の処理帯域の視点で見ると、並列度 1 のケースでは、赤丸が示す無圧縮時の帯域に対して青丸の LZ4 圧縮時の帯域は、SSD の物理帯域飽和により圧縮比の分だけ優位性があることが確認できる。一方、並列度 2 以上のケースでの”kblogrd” の処理帯域は、スレッド数が少ない状況では、ストレージの構成による差は見られず、スレッド数が増大する局面では、ストレージの構成によるばらつきは見られるものの LZ4 圧縮の有無で大きな差は現れていない。これらの結果から、LZ4 圧縮に伴う転送量 (転送時間) 削減により、CPU の処理帯域に対してストレージの物理帯域が不足する状況では性能改善に寄与してことが分かる。一方、CPU の処理帯域に対して潤沢なストレージ帯域がある状況では、展開処理等のオーバーヘッドにより”kblogrd” への CPU 時間割り当てが減少し、圧縮による転送量削減効果を相殺していると考えられる。また、テストデータの平均圧縮比は 1.51 倍であり、テストシステムのように圧縮に伴う大きな読み出し性能の劣化が無い条件であれば、LZ4 圧縮の導入で SSD の容量単価の緩和を期待できる。

1 年分のテストデータの読み出しが最も早かった構成は、SSD3 台で無圧縮の ZFS を構成したケースで、最大スレッド数である 6 スレッド時にピーク I/O 帯域 1090MiB/sec を達成し、全 1 年分の読み出しを 110 秒で完了している。NFS にて 1Gbps の全帯域が利用できたとしても約 10 倍速、現行システムの実効 NFS 性能を基準にすれば、20 倍速以上での読み出しに相当する。また、ベンチマークシステムは比較的古いシステムなので、最新世代の機材を利用すれば更なる改善を期待できる。

## 4. キャッシュサーバーの運用

前述のベンチマークで良好な結果が得られたので、SuperKEKB 加速器の実運用での改善を行なうべく、運転制御にて”kblogrd” を動作させている CPU サーバー 3 台中で最もメモリ搭載量の大きな 1U サーバーに Intel SSD 750 シリーズの 1.2TB モデル 1 機を導入し、KEKBLog アーカイブの読み出しキャッシュサーバーとして運用した。SSD を導入した CPU サーバーの諸元は、Table 2 の通りである。KEKBLog アーカイビングシステム

Table 2: Cache Server Specification

| | |
|---|---|
| Processor | Opteron 6174 2.2GHz 12core x2 |
| Memory | DDR3 128GiB(Quad Channel/Socket) |
| System Storage | 137GB HDD(mirror) |
| Cache Storage | Intel SSD 750 1.2TB x1 |
| OS | FreeBSD/amd64 10.2-STABLE |
| File System | ZFS |

本体への変更を最小限にするため、最新のアーカイブとの同期はキャッシュサーバーがNFS経由で収集サーバー上のログファイルの増分を取得することで行い、“kblogrd” による読み出しでは、ログ・インデックスファイルを open(2) する際にキャッシュ上のファイルの有無を確認し、キャッシュ上のファイルを優先的に使用するよう改修を行なった。

収集中のインデックス・ログファイルの差分同期では、一般的な rsync 等の同期ツールを NFS 越しに使用すると差分検出を行なう際に既存のファイル全域を走査するので、ログファイルのように比較的大きくかつ小さな追加書き込みが続く状況では効率が悪い。本件では、複製対象の更新パターンに特化した同期ツールをシェルスクリプトで新たに開発した。収集サーバーで収集中のファイルの更新を、ファイルサイズとタイムスタンプの変化で検知し、インデックスファイルとログファイルで別々の複製手法を適用している。更新頻度が低くファイルサイズが小さいインデックスファイルは、更新毎に cp(1) にて複製し、同一ファイルシステム上での mv(1) によるアトミックムーブでキャッシュを更新している。追記書き込みのみが頻繁に行なわれるログファイルは、複製開始時のソースファイルサイズを世代管理し、数世代前のファイル末尾から現在のファイル末尾までの内容を dd(1) によって一部上書きしながら複製している。末尾の一部を重複して複製しているのは、NFSv3 経由で収集サーバー上のログファイルを参照した際に、ファイルサイズの変更が確認された時点で、ファイル末尾付近にデータ書き込み前の 0x00 で埋まった状態が見える場合があるためである。ファイルサイズの変更からファイル末尾にデータ書き込まれるまでの時間には変動があるため、別途、cmp(1) によるログファイルの複製欠陥検査と必要な場合は再複製を 5 分毎に実施することで、キャッシュの整合性を維持している。SuperKEKB 加速器での Phase1 運転中のデータ収集量は平均 4MiB/min であり、末尾の重複書き込みによるオーバーヘッドを含めても SSD の書き込み帯域には十分な余力があるが、書き換え寿命を考慮し SSD への書き込み頻度を下げるために ZFS における書き込みトランザクションの最大メモリ滞在時間を標準の 5 秒から 30 秒へ延長している。

データ収集が完結したファイルについては、古くなったものから無圧縮の領域から LZ4 圧縮を適用する領域に移動し、SSD の使用領域を削減している。LZ4 圧縮による圧縮比は、平均で 1.42 倍となっている。

同期周期は、運用開始時は 30 秒間隔であったため、トレンドグラフ等では最大 30 秒の表示遅延が発生する問題があったが、複製プロセスの多重起動の管理や差分複製効率の改善により、大半の用途では問題の無い 2 秒間隔の同期を実現した。これ以上の表示遅延の短縮を行なうには、収集サーバーのローカルファイルシステムへの書き込みが NFS 上で可視となるまでの遅延時間やシェルスクリプト実行時の各種コマンドの fork/exec のオーバヘッド等の影響があるため、収集サーバーの実装を含めた改修が必要と思われる。

試験運用で用いた Table 2 の CPU サーバーの PCIe バスは第二世代であるため、SSD との物理帯域が制限されており、データ書き込み済みの状態での SSD からの連続読み出し帯域はおおよそ 860MiB/sec 程度となっている。また、”kblogrd” のプロセス当たりの I/O 処理能力は、200MiB/sec 程度である。データ収集サーバー及び NAS アプライアンスからの NFSv3 による読み出し帯域が、40MB/sec 前後で有るため、少数のデータの読み出し時 (単一プロセス時) は約 5 倍の高速化、複数グループからの同時読み出しや長期間の読み出し (並列動作時) では最大 21.5 倍の高速化が達成出来ている。また、当該 CPU サーバーの 128GiB のメモリの大半が、ZFS/NFS のキャッシュに利用され、複製直後のデータはほぼ全量がキャッシュヒットするため、短期のトレンドグラフでの読み出し時間が大きく改善した。

## 5. まとめ

試験的に導入したキャッシュサーバーは、SuperKEKB 加速器の Phase-1 運転中の KEKBLog 読み出し要求のほぼ全量を処理し、ユーザーが体感する読み出し遅延時間の軽減に大きく貢献した。また、収集中アーカイブファイルの同期に伴う読み出し可能となるまでの遅延時間増大も 2 秒程度に抑えることが出来た。KEKBLog アーカイブシステムへのキャッシュサーバー追加による読み出し時間短縮という目標は概ね達成された言える。

一方、現行の 1U サーバーと PCIe NVMe ストレージカードによる実装では、キャッシュ上に保存出来るデータは Phase-1 運転時の収集量から約 1 年分と推定される。本格的な SuperKEKB 加速器の運用で収集量が増えることを考慮すると、2U 等の大型筐体に複数の PCIe NVMe ストレージカードを搭載した大容量のキャッシュサーバーやアーカイブの種別毎に別のキャッシュサーバーを導入する等の施策が必要と思われる。

## 謝辞

KEKB 制御グループには、KEKBLog アーカイビングシステムのソースコード提供及び試験環境の構築あたり全面的な協力を頂きました。キャッシュサーバーの試験運用に参加頂いた SuperKEKB 運転員とコミッショニング参加者の皆様に感謝します。

## 参考文献

[1] Y. Ohnishi *et al.*, “Accelerator design at SuperKEKB”, Prog. Theor. Exp. Phys. (2013) 03A011; http://ptep.oxfordjournals.org/content/2013/3/03A011.full.pdf

[2] T. Kamitani, “SuperKEKB フェーズ 1（入射器＋リング）”, Proceedings of the 13th Annual Meeting of Particle Accelerator Society of Japan, Aug. 2016, MOOLP02

[3] http://www.aps.anl.gov/epics/

[4] T. T. Nakamura *et al.*, “DATA ARCHIVING SYSTEM IN KEKB ACCELERATORS CONTROL SYSTEM”, Proceedings of the 10th ICALEPCS, Geneva, Oct. 2005, PO1.077-7; https://accelconf.web.cern.ch/accelconf/ica05/proceedings/pdf/P1_077.pdf

[5] https://www.freebsd.org/ja/

[6] https://ja.wikipedia.org/wiki/ZFS