Hayabusa：高速に全文検索可能なログ検索エンジン

3.1 背景と目的

ネットワークやシステム運用の現場では安定したネットワーク運用やトラブルシュートを行うために、ネットワーク管理者がネットワーク機器から出力されるログを収集して統計情報として表示したり、トラブルの原因となるログを検索する方法が多く用いられています。また、セキュリティインシデントに対応する際にも、トラブル対応の場合と同様、ログからどのようなインシデントが発生したかを探ることがあります。

大規模なネットワークでは、多くのサーバ・ネットワーク・セキュリティ機器から通信を記録したログが日々大量に出力されるため、ネットワーク管理者はそれら大量のログをストレージシステムに蓄積し、高速にログ検索するシステムを管理しています。大規模なログ検索システムと蓄積システムを扱うためには、クラスタリングシステムや専用の管理ソフトウェアを用いることになり、本来はログの解析に時間を費やしたいところを、検索・蓄積システムの管理に時間が割かれることも少なくありません。

「ログの蓄積」と「ログの検索」がシンプルに動作する仕組みが、複雑なクラスタリングシステムを用いることなく実現できれば、ネットワーク管理者は蓄積・検索システムの管理に時間を割かれずに済み、本来の業務であるネットワークのトラブル対応やセキュリティインシデントの解析に集中することができます。

本稿では、多数のマルチベンダー機器が出力する大量のログを高速に蓄積でき、高速に検索可能なシステムとして実装したオープンソースソフトウェア「Hayabusa」について解説します。そしてシステムが扱うログの量が増加した場合には、シ

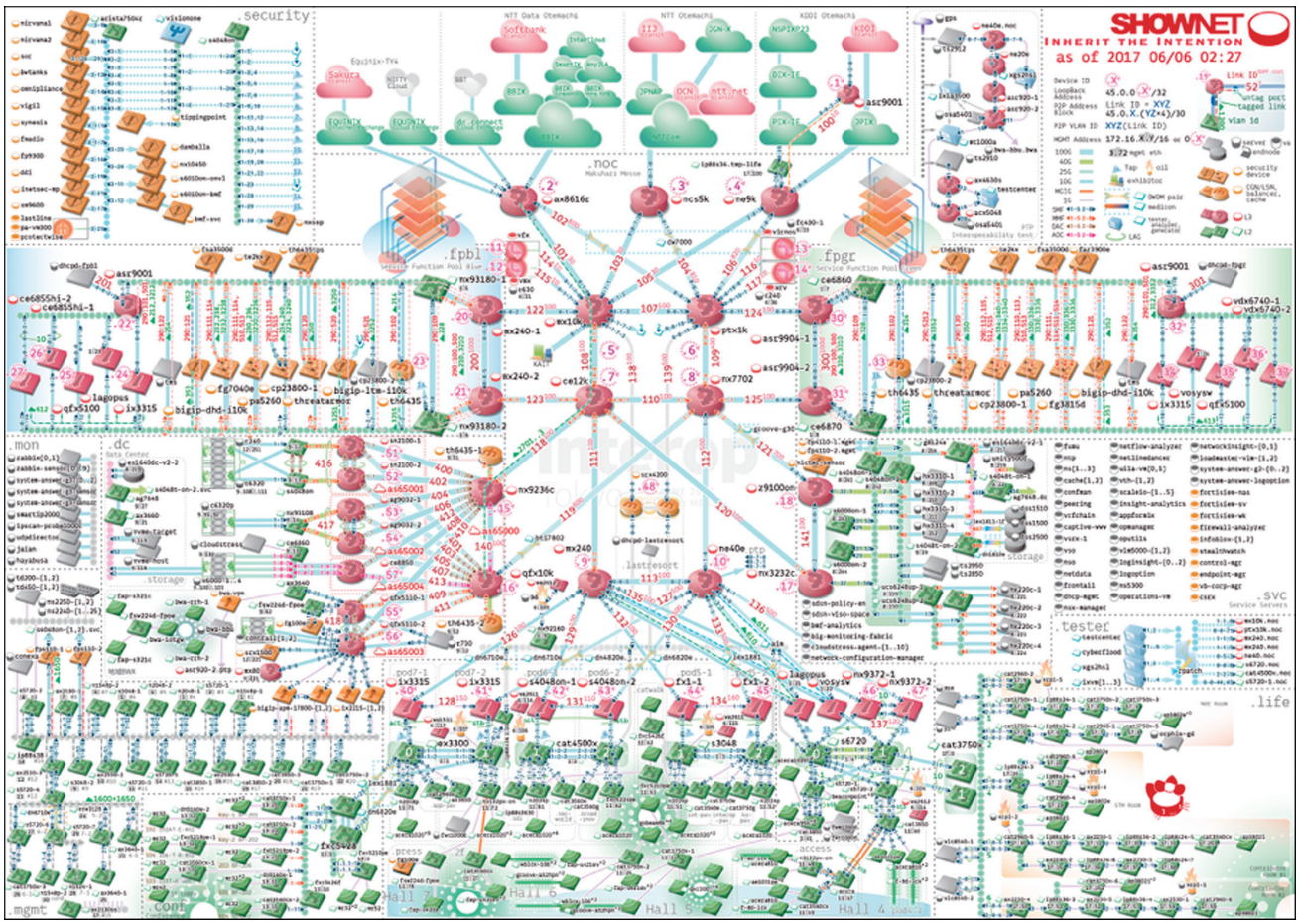

図-1 2017年のShowNetトポロジー図

ステムの検索性能が容易にスケールアウトすることで検索速度が飛躍的に向上する同ソフトウェアの分散システムのコンセプトモデルも併せて紹介します。

Hayabusaが実現する検索の高速化に関する評価は、Interop Tokyo[*1]で収集されたShowNetのsyslog実データを元に実験を行いました。この実データは、600以上のサーバ・ネットワーク・セキュリティ機器から出力されたものです。

3.2 ShowNet

ShowNetは、毎年幕張メッセで開催されるInterop Tokyo内で構築される相互接続検証とデモンストレーションを行う実験ネットワーク環境です。出展社へのインターネットアクセスをサービス提供する側面もあるため、実験とサービス提供の両方の側面を併せ持ちます。

図-1に示すように、ShowNetを構成する機材は数百を超え、実験ネットワークという性格上世界で初めて実ネットワークへと組み込まれる製品も多く、中には試作レベルの機材やソフトウェアもあります。

ShowNetを運用するメンバーは、これらの機材やソフトウェアを組み合わせて、サービス提供するネットワークを設計・構築して運用を行います。安定的にシステム構築を行うために、機材から出力されるsyslogを収集分析し、システムのバグやエラー、構成の成否を判断する運用が行われます。ShowNetを構成する機材は多岐にわたり、更に最新鋭のソフトウェアやハードウェアが大量に投入されるため、機材やソフトウェアを管理する運用メンバーは機材からどのようなログが出力されるかを事前に知ることは困難です。

ShowNetでは、監視システムの1つとしてsyslogを収集し分析するシステムが運用されます。ほぼすべての物理機器・仮想機器群からsyslogが送信され、そのsyslogを集約し検索するシステムが構築されます。過去には、秒間２万件以上のログが飛び交ったり、1日に2億件以上のログが蓄積されたこともありました。

特定のソフトウェアやハードウェアからのログであれば、ログフォーマットを推測することも可能ですが、ShowNetを構築する機材には同一のソフトウェアやハードウェアは少なく、かつ大量のメッセージが最新のファームウェアやソフトウェアのデバッグ情報として出力されるため、一般的なログ蓄積・解析ソフトウェアでリアルタイムに統計を表示させたり検索を行うことは非常に難しいです。

3.3 Hayabusa

Hayabusa[*2]は、ShowNetで収集された大量のsyslogを高速に検索するためのシステムとして設計されました。図-2にHayabusaのアーキテクチャを示します。

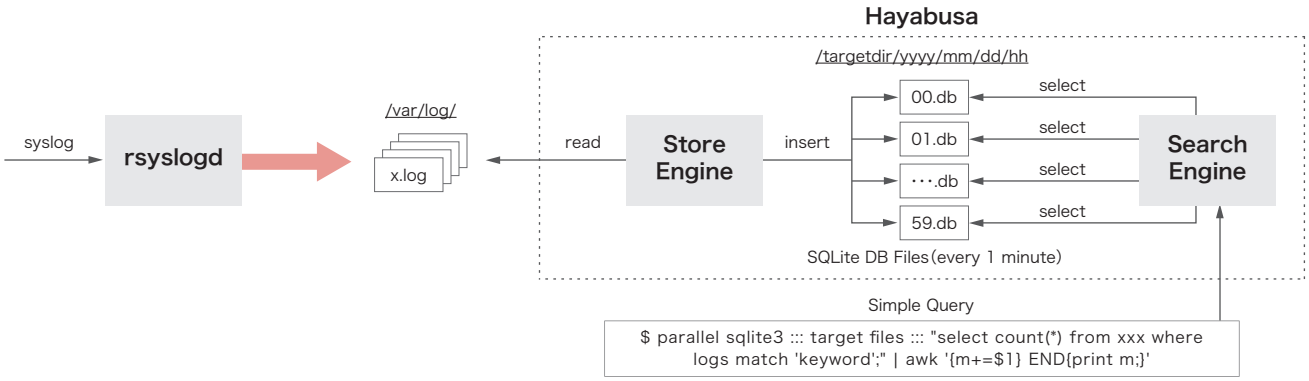

図-2 Hayabusaのアーキテクチャ

＊1　Interop Tokyo(https://www.interop.jp/)。
＊2　Hayabusa(https://github.com/hirolovesbeer/hayabusa)。

Hayabusaはスタンドアロンサーバで動作し、CPUのマルチコアを有効に使って高速な並列検索処理を実現します。Hayabusaは大きくStoreEngineとSearchEngineの2つのシステムに分割されます。StoreEngineはcronにより1分ごとに起動され、ターゲットとなるログファイルを開き、ログファイルに記録されたログメッセージをSQLite3[*3]ファイルへと変換します。ログデータは1分ごとのSQLite3ファイルへと分割され、検索時に複数プロセスにより並列処理されます。ログが保存されるディレクトリは以下のように時間を意味する階層として定義されます。

/targetdir/yyyy/mm/dd/hh/min.db

上記のディレクトリ階層は、以下の意味を持ちます。

/ターゲットディレクトリ/年/月/日/時間/分ごとのSQLite3ファイル

このように定義することで、ログ検索のための時間情報をデータベース内部に保持することなくディレクトリとのマッチングで行うことができ、時間のクエリ条件を指定することなく時間指定のログ検索が可能になります。ログが保存されるSQLite3ファイルはFTS（Full Text Search）と呼ばれる全文検索に特化したテーブルとして作成され、高速なログ検索を実現します。SearchEngineは、並列検索性能を向上させるために分単位に細分化されるFTSフォーマットで定義されたSQLite3ファイルへアクセスを行います。各SQLite3ファイルへはGNU Parallel[*4]を用いて並列にSQL検索クエリが実行され、結果はUNIXパイプラインを経由してawkコマンドやcountコマンドを用いて集計されます。

Hayabusaはスタンドアロン環境で動作しますが、小規模なApache Spark[*5]のクラスタよりも全文検索性能が高く、図-3で示すように、先行研究[*6]では3台のApache Sparkクラスタより27倍速い検索性能を示しました。

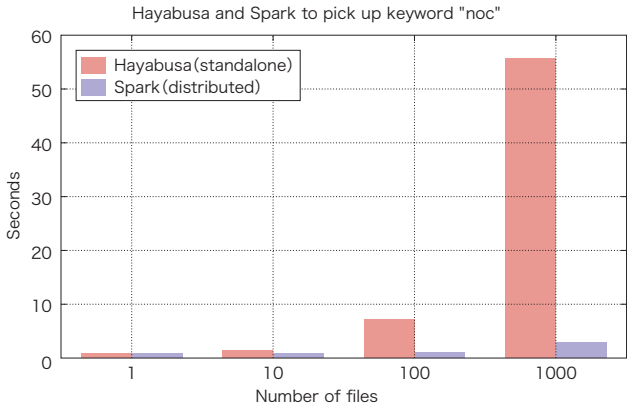

図-3 HayabusaとApache Sparkとの性能比較

＊3　SQLite3（https://www.sqlite.org/）。
＊4　GNU Parallel（https://www.gnu.org/software/parallel/）。
＊5　Aapache Spark（https://spark.apache.org/）。
＊6　H. Abe, K. Shima, Y. Sekiya, D. Miyamoto, T. Ishihara, and K. Okada. Hayabusa: Simple and fast full-text search engine for massive system log data. In Proceedings of the 12th International Conference on Future Internet Technologies, CFI'17, pages 2:1?2:7, New York, NY, USA, 2017. ACM.

3.4 Hayabusaの分散処理

しかしながら、スタンドアロン環境にはハードウェア性能の限界が存在し、いつかは規模が拡大した他の分散処理クラスタに性能を抜かれてしまうと推測されます。そこでHayabusaは、スタンドアロン環境という制約を取り払い、複数ホストでHayabusaの分散処理環境を構築し、検索性能がスケールアウト可能なアーキテクチャの実現を目指しました。

スタンドアロンで動作するHayabusaを分散処理システムとして再定義し、検索処理性能をスケールアウトさせる実験を行いました。Hayabusaのスタンドアロン処理性能を活かすべく、処理リクエストを高速なRPC(Remote Procedure Call)としてクライアントから処理対象ホストへと送り込みます。GNU Parallelを用いた並列検索を処理ホスト内で実行し、結果はRPCを用いてクライアントへと返し集計を行います。これによりHayabusaが本来実現していた高速なスタンドアロン環境での検索と、RPCによる高速なリクエスト/レスポンスを実現することができます。分散ホスト環境においてスケールアウトする検索機能を実現するために、データの並列蓄積と分散検索に分けて設計を行いました。

3.4.1 並列蓄積と分散検索

検索処理をスケールアウトさせるために、どの処理ホストに処理リクエストが届いても検索可能な状態になるように、今回の実験ではすべての処理ホストに同一のデータを保持させる方法を用いました。これはすべての処理ホストへとsyslogデータを複製して配送することを意味します。これにより、どの処理ホストへ処理リクエストが渡ろうとも同じ結果が返ることが保証されます。

分散している処理ホストへの検索リクエストの実現にはRPCを用いました。Hayabusaの分散処理では、クライアントからのリクエストを受け取った処理ホストは、スタンドアロンのHayabusaと同等にローカルディスクに保存した複数のデータベースファイルに検索クエリを発行することを前提とします。SQLite3のコマンドを検索リクエストのパラメータとして処理ホストに受け渡すことも可能ではありますが、本提案ではスタンドアロンのHayabusaの性能を活かしつつ、シンプルに分散処理を実現するために、スタンドアロン環境で処理されるものと同等のGNU Parallelのコマンドをリクエストのパラメータとして受け渡します。分散検索では複数の処理ホストへと処理リクエストが発行されることから、クライアントが処理リクエストをキューイングし、処理ホストで動作するWorkerがキューイングされた処理リクエストを取得して結果を返すProducer/Consumerモデルを選択しました。これにより、各処理ホストで動作するWorkerは処理リクエストを取得した後に、検索処理の実行が可能となります。

本提案では、すべての処理ホストにデータが複製されていることから、どの処理ホストへ処理リクエストが渡ったとしても同様の処理結果が返されます。また、クライアントからの処理リクエストが特定の処理ホストに集中しないようにロードバランシングを行い、各処理ホストへ均等にリクエストを配布可能にしました。

3.4.2 実装

■ 並列蓄積

syslogをすべての処理ノードへ複製するために、本研究ではオープンソースソフトウェアであるUDP Samplicator[*7]を利用しました。syslogの複製イメージを図-4に示します。UDP Samplicatorは送信元アドレスを変更せずに受信したUDPパケットを指定した対象ホストへ転送することができます。これにより転送先のホストは、あたかも自身が送信元から直接データを受信したかのようにUDPパケットを受信することができます。

通常、UDP Samplicatorは1プロセスでUDPの転送処理を行います。しかしながら大量のsyslogを受信した場合には、プロセスの負荷が上昇しCPUのコア利用率が100%になってし

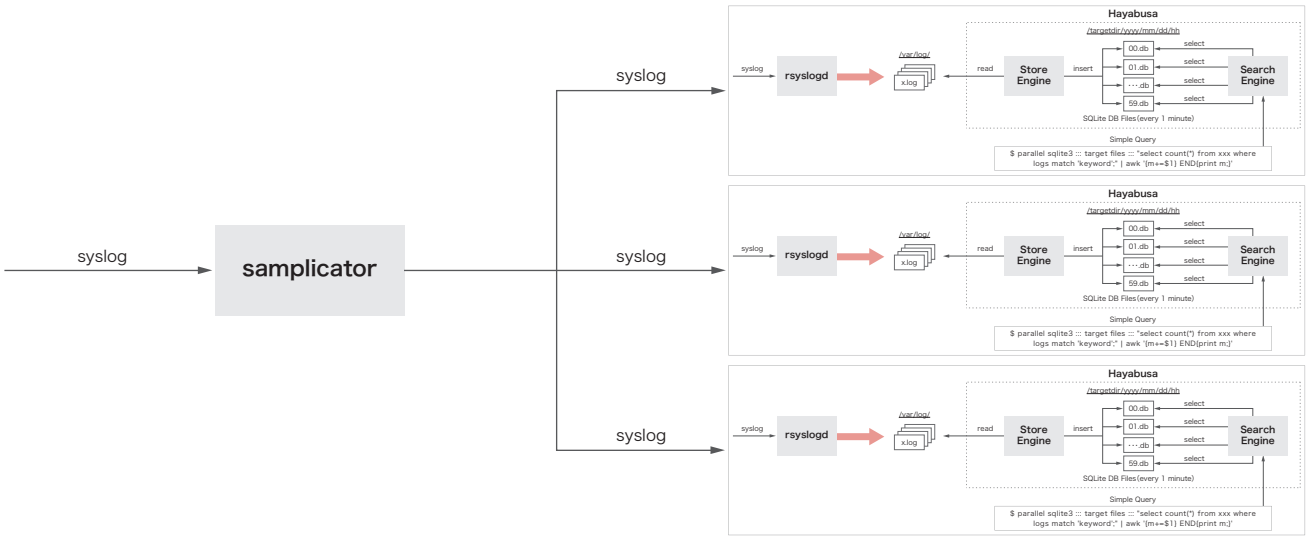

図-4 UDP Samplicatorを用いたsyslogの複製

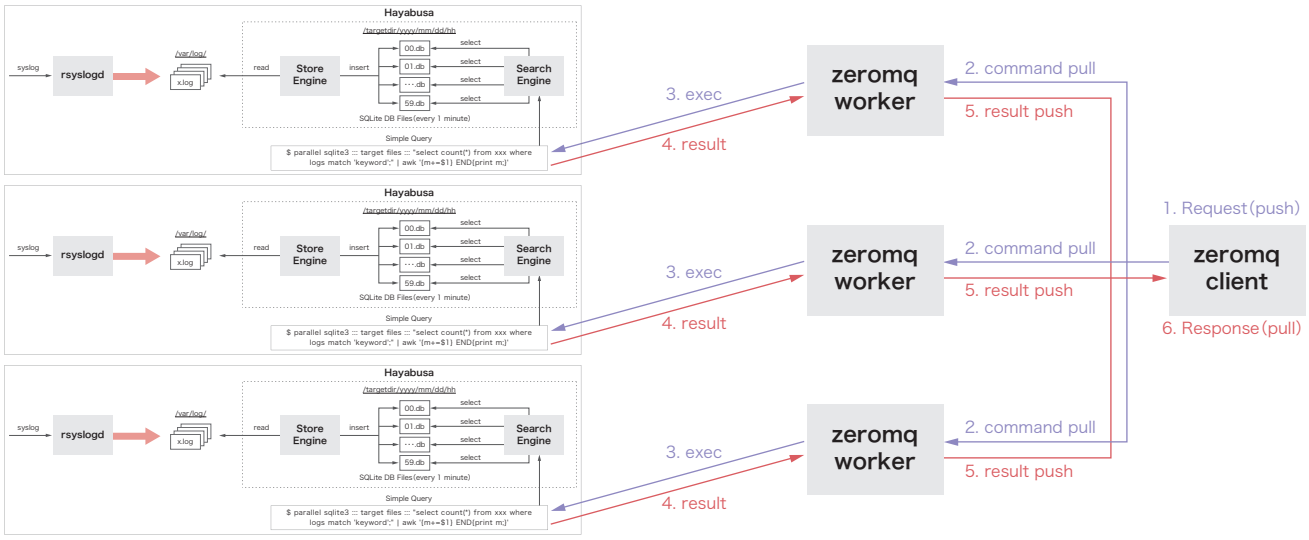

図-5 UDP Hayabusaで用いるZeroMQのPush/Pull実装

*7 UDP Samplicator(https://github.com/sleinen/samplicator)。

まう場合があります。その結果、パケット転送処理が追いつかなくなり、データが破棄される可能性が生じます。そこで本実験ではUDP Samplicatorにパッチを当て、マルチプロセスとして動作するようにソースコードの修正を行いました。具体的には、socketのオプションとして「SO_REUSEPORT」を追加することにより、複数プロセスで同じ待ち受けポートが利用可能となります。本提案の場合には、syslog受信ポートである「UDP 514ポート」が複数プロセスで共有されることとなり、UDP 514ポートへと届いたパケットは、複数のUDP Samplicatorプロセスに自動的にロードバランスされます。これにより大量にsyslogを受信した際でも、1CPUコアではボトルネックになりがちなsyslogパケットの複製と転送をCPUコアスケールすることが可能となります。

■ 分散検索

Producer/Consumerモデルは多くのソフトウェアで実装可能ですが、本研究ではRPC処理が高速に実行可能で、ライブラリを用いることでクライアントとWorkerプロセスを実装可能なZeroMQ[*8]を用いました。ZeroMQは高速に動作する分散メッセージキューとして利用され、「Request/Response」「Publish/Subscribe」「Push/Pull」などたくさんのメッセージングパターンを容易に実装することができます。本提案では、「Push/Pull」パターンを用いてProducer/Consumerモデルを実装しました。

図-5で示すように本提案でのクライアントは、PushとPullの2つの役割を持つように実装しました。これによりリクエストの発行からキューイング、結果の取得と集計をクライアント1プロセスで行うことができます。

3.5 評価

分散システムとしてのHayabusaのスケールアウト性能を調べるため、処理ホストが増加した場合に処理時間が短縮できるかどうかの試験を行いました。処理ホストは1台から10台の範囲で増加し、クライアントは1日分のデータに対して繰り返し100回リクエストを実行します。100回分のリクエスト対象のレコードサイズは、144億レコード(1分間のsyslog流量は10万件と仮定)となります。

図-6に示すように、ホスト1台のときの検索処理時間は約468秒でしたが、ホストの台数を増やすに従いホストの数で割った時間に近づき、10台のホストでかかる処理時間は約10分の1である48秒になりました。

また、秒間に処理可能なレコードのスキャン数はホスト1台の場合には3000万件ですが、ホスト数が10台の場合には10倍である3億件まで増加します。これはホストを1台から10台まで増加させた場合、検索の処理性能がホストの台数に応じて意図したとおりリニアにスケールアウトしていると言えます。

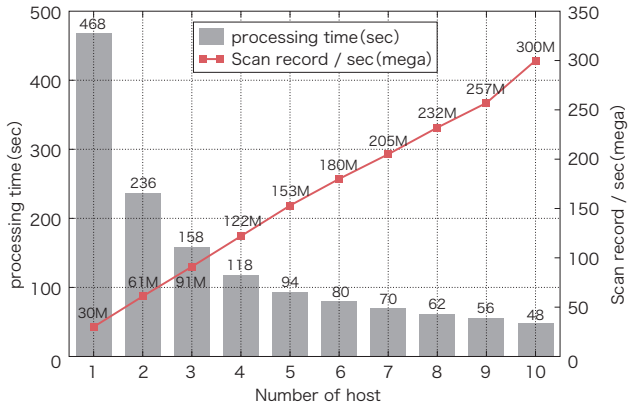

図-6 検索ホストのスケールアウト性能

*8 ZeroMQ(http://zeromq.org/)。

次に10台のホストで処理を実行する場合にWorkerの数を1プロセスから16プロセスの間で増加させました。16という数字の根拠は本実験に用いたサーバのCPUの物理コア数であり、物理コアの数だけスケールアウト可能かどうかを確認する意図があります。10台のホストで1Workerプロセスのみを動作させた場合にかかった処理時間は約48秒でしたが、10台のホストで10Workerプロセスを動作させた場合の処理性能は最高6.1秒となり、処理速度が約8倍高速化しました。10Worker以上の処理速度は横ばいとなったため、16プロセスで最大値を出すためには何かしらの工夫が必要と思われます。

ホスト数とWorker数の両方のスケールアウトを組み合わせた結果、1台の処理ホストで1Workerプロセスを用いた場合で約468秒かかっていた検索時間が、10台で10Workerプロセスを動作させた場合では6.1秒まで短縮し約78倍の高速化が実現しました。

3.6 今後の課題

本研究では検索性能を向上させる分散クエリに対応するため、各処理ホストに同一のsyslogデータを複製する手法を用いました。これは、本質的には重複するデータを大量に複製する行為であり、データの量が増加すればするほどネットワーク帯域と保持するデータに無駄が発生することを意味します。Hadoop[*9]のHDFSのようにレプリケーション数を設定し、複数ホストでデータを分散させ保持することはもちろん可能ではありますが、その場合にはメタデータ管理機構でデータの管理を行い、データアクセスはメタデータ管理機構経由となり、ストレージへのアクセス性能を低下させ、検索自体の処理性能も低下させる恐れがあります。Hayabusaと分散ストレージの利点欠点を考慮し、Hayabusaに適した分散ストレージを実現することが1つの大きな課題となります。

またHayabusaは検索基盤システムとして動作するため、その上で動作する具体的なアプリケーションソフトウェアを組み合わせることでさらなる有益なシステムとなり得ます。

syslogを高速に検索できることから、先行研究であるインベントネットワークにおけるsyslogを用いた異常検知[*10]と組み合わせることで、高速に動作する異常検知アプリケーションを作成することが可能になると考えます。

＊9　Apache Hadoop（http://hadoop.apache.org/）。
＊10　阿部博 and 敷田幹文. イベントネットワークにおけるsyslogを用いた異常検知手法の提案と実でータを用いた評価. In インターネットと運用技術シンポジウム2016論文集, volume 2016, pages 57-64, dec 2016.

3.7 Hayabusaの応用

セキュリティ分野の研究では、攻撃の予測や検知を行うことが目標の1つとされます。

増加するサイバー攻撃に対抗するために、攻撃の兆候をリアルタイムに分析し、発生し得る攻撃とその深刻度、影響範囲を予測することが可能になれば、今まで担当者の能力で担われていたセキュリティ事案に対する属人的な対応を、機械学習や深層学習を用いてサポートすることが可能となります。

IIJ技術研究所は、東京大学、東京工業大学、奈良先端科学技術大学院大学と連携し、「サイバー脅威ビッグデータの解析によるリアルタイム攻撃検知と予測」[*11]プロジェクトへ参加しています。本プロジェクトでは、攻撃の兆候をリアルタイムに分析する基盤として、その一部にHayabusaが組み込まれます。更に、他のデータ解析ソフトウェアや機械学習ソフトウェア(R、Chainer、Pandasなど)と連携をして、攻撃の兆候を統計や機械学習で分析する基盤として動作します。

3.8 まとめ

本稿では、オープンソースソフトウェアであるHayabusaの説明と分散処理による評価について解説しました。評価の対象となるレコード数は144億レコードであり、144億レコードを約6秒でフルスキャンできたということは、GoogleのBigQueryに匹敵するデータのフルスキャン速度が実現できたことを意味します。10台の処理ホストでこれほど高速なスキャンを実現できるということは、コスト面でもリーズナブルで高性能な分散処理システムであると言えます。

なお、本稿のより詳細な解説は論文[*12]として公開されています。

本研究の一部は、国立研究開発法人科学技術振興機構(JST)の研究成果展開事業「戦略的創造研究推進事業(CREST) JPMJCR1783」の支援によって行われています。

執筆者:
阿部 博(あべ ひろし)
IIJ 技術研究所 研究員。

執筆者:
島 慶一(しま けいいち)
IIJ 技術研究所 主幹研究員。

*11 「サイバー脅威ビッグデータの解析によるリアルタイム攻撃検知と予測」(https://www.jst.go.jp/kisoken/crest/project/1111094/1111094_13.html)。
*12 阿部博 and 篠田陽一. スケールアウト可能なログ検索エンジンの実現と評価. In インターネットと運用技術シンポジウム2017論文集, volume 2017, pages 73-80, nov 2017.