ログ解析自動化による業務効率化

○田中昌二

岐阜大学情報連携統括本部

概要

情報システムや情報ネットワークの管理において、各機器が出力するログは、システム障害やインシデント発生時の原因究明に必須となるのはもちろんのこと、各システムの運用状況を監視するための非常に重要な情報である。しかし、日々出力されるログのサイズは非常に大きく、こうしたログに記録される異常の兆候を、担当者の目視によりチェックすることは非常に困難である。そこで岐阜大学情報連携統括本部では、ログ収集サーバに各種ログを集約し、収集されたログを定期的に、あるいはリアルタイムで自動解析する仕組みを構築した。また、解析結果の可視化や、異常検知時のアラート通報なども自動化することで、業務量やヒューマンエラーの低減、システム異常発生時のレスポンス迅速化を実現した。

1　はじめに

サーバやネットワークスイッチなどの情報関連機器は、日々大量の動作記録（ログ）を出力しており、こうしたログの解析は情報システムや情報ネッワークの運用・保守・維持・管理業務において非常に重要なファクタとなっている。

最も一般的なログの利用目的に障害やインシデントといったトラブルの発生時の原因調査があり、記録されたログを解析することでシステムやサービスの異常な動作を把握することが可能である。さらに、ログにはトラブルとして顕在化していない問題の兆候が含まれている場合があり、これらを適切に監視することにより、そうしたトラブルを未然に防止し、もしくはその被害を最小限に留めることが可能となる。

別視点でのログ利用目的として、システム利用状況の統計取得が挙げられる。情報システム部門以外の構成員に対して情報システムのおかれた状況を説明するためには、その情報システムが実際にどの程度利用され、その状況が日々どのように変化しているかを示す必要がある。こうした場合には、より効果的に必要な情報を伝えるために、ログの集計結果をグラフなどで可視化することも重要となる。

このように非常に重要な業務でありながら、情報関連機器が出力するログの特性上、担当者の目視によるログチェックは困難な点も多い。そのため岐阜大学情報連携統括本部では、それらの一部を自動化することで業務の効率化を行った。本稿では、そうしたログ解析業務の自動化事例と、その導入効果について述べる。

2　ログ解析上の問題点

ログ解析業務には大小さまざまな課題が存在するが、代表的なものとしては以下に示す３点が挙げられる。

2.1　ログサイズ

岐阜大学情報連携統括本部では主な全学向けサービスとして、電子メールサーバや統合認証サーバなど数十台のサーバからなる「キャンパス基幹情報システム」と、ネットワークインフラを構成するネットワークスイッチや次世代型ファイアウォールなど数百台の機器からなる「キャンパス情報ネットワーク」の 2 システムを運用している。これらのシステムを構成する機器からは、合計で 1 日にファイルサイズにして 50GB 以上、行数では 3 億行を超えるログが出力される。このように、1 秒間に 3000 行以上が記録され続けるロ

グを人間が目視で監視することは明らかに不可能である。

2.2　ログフォーマット

ログに記録される情報のフォーマットは、そのログを出力するシステムやソフトウェアによって様々である。出力される情報の種類に違いがあるのはもちろんのこと、ログに必ず記録されるタイムスタンプですらそのフォーマットは統一されていない。以下に示す 4 つの例はすべて同じ時刻（日本時間 2019 年 3 月 1 日 15 時 20 分 00 秒）を表現するタイムスタンプだが、これらのうちどれが使用されるか（場合によってはさらに別のフォーマットである可能性も存在する）はログを出力する機器やソフトウェアによって異なる。

1. Mar 1 15:20:00

   UNIX 系 OS で一般的に利用されている syslog[1]の標準フォーマット。英語圏の一般的な時刻表記に近く人間にとって可読性は高いが年情報やタイムゾーンが記録されない。ログ解析を行う際にはそのログが何年時点のものであるかを別に管理する必要があるほか、タイムゾーンが異なる複数サーバ上のログを比較する際には時差を補正する必要がある。また、月情報が英語略称であるため、単純に文字列として比較した際に、時系列順に並ばないという問題もある。
2. 2019-03-01T15:20:00+09:00

   ISO8601 で国際的に標準化された時刻フォーマット。タイムゾーンを含め必要な情報がすべて記載されており、人間による可読性も高い。一方、プログラム上で 2 つのタイムスタンプ間の経過時間を知りたい場合などは、一旦、次項に示すエポック秒へ変換するなどの処理が必要となる。
3. 1551421200

   UNIX エポック（世界標準時 1970 年 1 月 1 日 00 時 00 分 00 秒）からの経過秒数（エポック秒）により時刻を表現する方式。タイムゾーンによらない表現が可能であるほか、時刻の比較を数値として高速に実行可能であるためコンピュータにとっては扱いやすいフォーマットであるが、可読性は著しく低い。
4. @400000005c78cf1a00000000

   国際原子時（TAI: Temps Atomique International）の名を冠し、マイクロ秒（100 万分の 1 秒）まで表現可能な TAI64N フォーマットによる表記。こちらも UNIX エポックからの経過秒数が記載されているが、整数部、小数部それぞれが 16 進数表記となっており。可読性は 3.と比較してもさらに低い。

これらのうち、1.や 2.の形式であれば、比較的容易に目視で日時を識別可能であるが、3.や 4.の形式では何らかのツールを利用しない限り目視での識別はハードルが高い。場合によっては、タイムスタンプの記録形式が異なる複数のログファイルを横断的に解析する必要があり、目視の場合それぞれのファイルに含まれるログが同一時刻に出力されたかどうかを判断することすら困難である。

また、通信関係のログには必ず登場する IP アドレスも目視による監視を困難にする要素の１つである。一般的に IP アドレスが近い接続元は同じ組織もしくは少なくとも同じ国からのものであると想定されがちだが、実際には末尾が 10 違うだけの 2 つの IP アドレスが、直線距離で 2000km 離れた国に割り当てられている事例もある。

このほか、1 つの処理がログ 1 行で完結せず、何らかの識別符号で複数のログを突き合わせなければ状況を把握できない情報も存在し、さらにそれらが複数並行して記録されるに至っては、単一のログファイルに記録された内容の把握ですら容易ではない。

2.3　ログの解析によって生じる負荷

前述したサイズ、フォーマットは主に人がログ解析をする際に顕在化するが、ログ解析によって生じる負荷の問題はプログラムを用いてログ解析を自動化した際に顕在化する。ログ解析を自動化した場合、そのことによって CPU 時間、主記憶容量（メモリ）、補助記憶容量（HDD/SSD など）、IO（入出力）などといった計算資源が消費される。

電子メールサーバや Web サーバなど、ログを記録する装置が汎用サーバである場合、同サーバ内でこうしたプログラムを用いたログ解析を行うことも可能であるが、そうすることによってサーバ本来の機能である電子メール配信や Web コンテンツ公開に支障をきたすような状況が起きてはならない。特に大きなログファイルを対象とする、もしくは複雑な操作を伴うなどといった所謂「重い」処理については、実サービスを提供する情報機器以外で行うことが望ましい。

3　ログ収集環境

一般的に、ログファイルは出力する各機器に搭載された補助記憶装置に保存される。Linux などの汎用サーバであれば、当該機器の内部でログ解析の仕組みを実行することも可能であるが、前述の通りログ解析が実サービスに悪影響を及ぼすことは避けねばならない。また、ネットワークスイッチなど特殊用途の専用機器については、補助記憶装置の容量も少ないうえ独自の処理を実行すること自体が事実上不可能である。そこで、岐阜大学情報連携統括本部ではキャンパス基幹情報システムおよびキャンパス情報ネットワークに対して、それぞれの要件に応じたログ収集環境を構築した。図 1 にログ収集環境の接続構成を示す。

3.1　キャンパス基幹情報システム用ログ収集サーバ（gusyslog）

キャンパス基幹情報システムを構成する機器は、そのほとんどが全学仮想基盤上で動作する仮想サーバである。出力される個々のログは、多いものでも 1 日あたり数百万行程度であるが、基幹情報システム全体で数十台のサーバが稼動しており、それら全体の総量は決して小さくない。

複数機器で記録されるログを収集する方法としては、syslog プロトコルを利用して、ネットワーク越しにログを転送する方式が一般的であるが、以下のような理由から gusyslog ではその方式を採らず、ネットワークマウントしたログ収集用 NFS 領域を介してログの収集を行う方式を採用した。

1. 全学仮想基盤が学外データセンタに設置されているのに対して、機器間の通信制御は岐阜大学のキャンパス内に設置されたネットワーク機器が行っており、場合によってはデータセンタと岐阜大学キャンパスとの間に無駄な通信の往復が生じる可能性がある
2. syslog では、ログを facility（種類）と severity（重要度）で分類し、保存するファイルなどの振り分けを行うが、ログを出力するサーバの台数が多く、また 1 台のサーバが複数のログを出力するため、すべてのログを個別に処理できない

まず、全学仮想基盤のバックエンストレージとしても利用される大学基幹ストレージ上に、2TB のログ収集用ボリュームを用意した。ログを出力する各サーバはそのログ収集用領域を NFS マウントし、日次処理として前日までの各種ログファイルを圧縮・アーカイブする。さらに、ログ解析専用のサーバをやはり仮想基盤上に構築し、同領域を読み込み専用でマウントすることで、当日分を除く全サーバのログに一括してアクセス可能とした。実サービスを提供する各サーバ上では、記録された当日のログに対してリアルタイムに処理が必要な最小限の工程のみを実行することで、ログ解析が実サービスへ与える影響を最小化している。

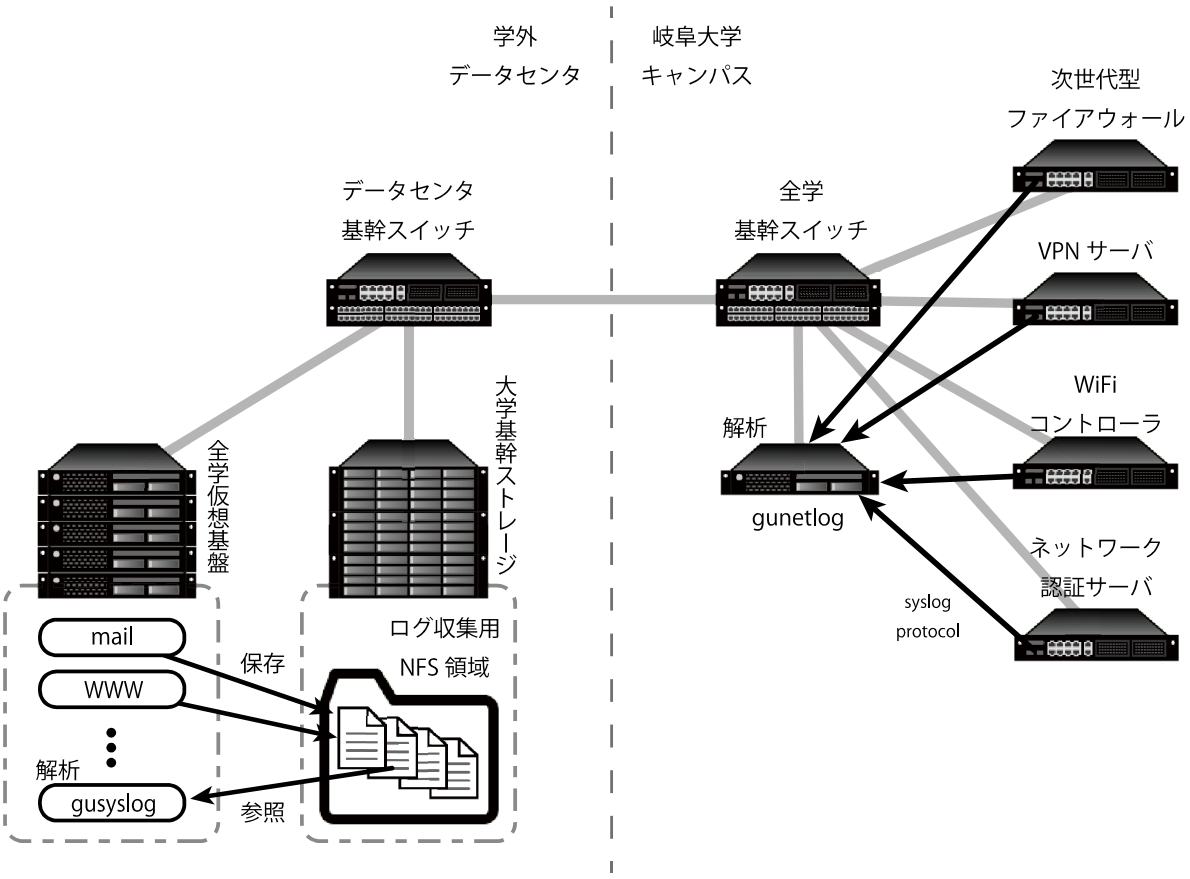

図１　ログ収集環境接続構成

また、全学仮想基盤、大学基幹ストレージ共にデータセンタに設置されており、さらに各サーバによるログ収集用領域のマウントにも専用のプライベートネットワークを利用することで、ログ収集に伴う通信をデータセンタ内で完結させている。

3.2　キャンパス情報ネットワーク用ログ収集サーバ（gunetlog）

gunetlog 側のログ収集機構では、以下のような理由から、gusyslog のような NFS 領域を介した方式ではなく、標準的な syslog プロトコルを利用する方式採用を採用した。また、gunetlog そのものも全学仮想基盤を利用せず、独立した物理サーバを利用する構成となっている。

1. キャンパス情報ネットワークを構成する機器は、ネットワークスイッチや次世代型ファイアウォールなど、特殊な用途の機器がほとんどを占めており、そうした機器は利用可能なストレージ容量が限られるため、内部ストレージに１日分のログを保存できない恐れがある
2. これらの機器はその能力のほぼすべてをハードウェア実装された特殊機能に費やしているため、ログの圧縮やアーカイブといった機能も持っておらず、gusyslog と同様のログ収集が困難である
3. ネットワーク機器のほとんどが岐阜大学キャンパス側に設置されており、syslog サーバを全学仮想基盤上に構築した場合、岐阜大学キャンパスとデータセンタ間の通信帯域を、転送されるログデータが圧迫する恐れがある
4. 万が一、岐阜大学キャンパスとデータセンタ間の回線に障害が発生した際に、岐阜大学キャンパスに設置されたネットワーク機器に関するログが取得不能になる恐れがある

ネットワーク経由でログデータを受け入れる gunetlog には 1000Base-T（1Gbps）のネットワークインタフェースを２つ搭載し、全学基幹スイッチと計 2Gbps で直接接続している。なお、gunetlog は利用者向けの実サービスを提供するサーバではないため、ログの収集だけでなく解析も含めてその上で行う。

4　開発環境

ログの各行から特定カラムを内容ごとに集計するといった単純な処理であれば、awk[2]や grep[3]、sort[4]、uniq[5]といった UNIX の標準的なコマンドを組み合わせるだけで実行可能な場合もある（図２）。しかし、その内容に応じたグルーピング、正規化、補完などより複雑な処理が必要な場合には、何らかのプログラミング言語を利用することになる。ほぼすべてのプログラミング言語はテキストデータの入出力機能をもっており、テキスト解析に利用することが可能であるが、特に、以下のような条件を満たした言語はログ解析に適しているといえる。

1. 部分文字列の抽出や特定の区切り文字での分割、複数文字列の結合、指定パターンによる置換といったテキスト処理に有用な機能を備えていること。さらに、正規表現による文字列のマッチングなどをプログラミング言語の標準機能として備えていることが望ましい。
2. 開発の容易さ。プログラミング言語をサーバへ導入する際のハードルの低さや、作成したコードの実行しやすさ、プログラムコードの読みやすさなどが含まれる。インシデント発生時の突発的なログ解析などは、十分な設計・検証を行う時間的な余裕があることは稀であり、トライアル・アンド・エラーで情報を整理する場面において非常に重要。
3. 実行速度が高速であること。前述の通り、ログデータは通常のテキストデータと比べると量が多く、集計や統計といった処理に多くの時間を要する。このとき、ログ 1 行当たりの処理速度が 10 万分の 1 秒短くなれば、2 億 4000 万行のログファイルを処理する時間を 40 分短縮可能である。

Perl[6]や Ruby[7]、Python[8]などのスクリプト言語は、正規表現を含む強力なテキスト処理機能を有しており、またインタプリタ言語であるため実行にあたってコンパイルやリンクといった手順が不要である。さらに、多くの UNIX 系ディストリビューションに標準で搭載されているか、もしくは標準的なパッケージマネージャを通じて導入可能な場合が多く、これらの言語は総じてログ解析に適しているといって良い。

なお、岐阜大学情報連携統括本部ではログ解析の多くの場面で Ruby を採用している。Ruby は国産のオブジェクト指向スクリプト言語であり、上記に挙げた特徴に加えオブジェクト指向の特徴であるコード再利用面などでのメリットを享受できる。ただし、このことはオブジェクト指向言語であれば Python などでも同様のことが言えるため、Ruby が特に優れているというわけではない。使用言語の選択は、担当者の習熟度と技術継承を考慮して決定されることが望ましい。

また、特にサイズの大きなログを対象とする場合や、リアルタイム処理が必要な場合など、実行速度が求められる場面では、プログラミング言語 Crystal[9]を採用した処理もある。Crystal は Ruby と非常によく似た構文を持ちながら、静的な型システムと高速なネイティブコードへのコンパイルが可能なプログラミング言語である。基本的にはコンパイル型言語であるがコンパイルと実行を 1 コマンドで行う機能を持っており、インタプリタ言語のように手軽に実行できる。また、構文面での Ruby との類似性は、単純な処理であれば、Ruby と Crystal の両者で実行可能なコードを記述可能なほどであり、Ruby 経験者であれば習得は容易である。実際、Ruby から Crystal への移植により処理時間が 1/3 以下に短縮された事例もあり、処理の高速化には非常に有効である。

```
# cat maillog | grep " connect from " | awk '{print $NF}' | sort | uniq -c
```

図２　当日のメールログから送信元サーバを集計する例

5　事例１）リアルタイム異常検知

外部からの不正アクセスパターンの 1 つに、何らかの方法で入手した正規利用者の認証情報を用いて、正規の認証手順を踏んだ上で大量の不正メール（迷惑メールやフィッシングメールなど）を送信するという事例がある。この不正アクセスの被害にあった場合、大学側が不正利用の被害者であるのみならず、外部に対して不正メールを送信した加害者（少なくとも不正メールの受信側からはそう見える）となってしまう点で迅速な対応が求められる。

特に近年では、攻撃者側がボットネットを利用して電子メールの分散大量送信が可能となったことで、1 時間に 1 万通を超える不正メールを送信されてしまう事例もあり、1 秒でも早い不正アクセスの検知と対応が必要である。しかし、不正メールの送信であっても 1 通 1 通は正規の認証を経るため、電子メールサーバソフトウェア上ではそれを正規利用者による送信と区別することはできない。この問題に対応するため、岐阜大学情報連携統括本部では gusyslog を中心として図３に示す仕組みを構築した。

なお、gusyslog が参照可能なログ収集用 NFS 領域は、通常、前日までのログアーカイブのみが保存されるため、オンタイムのログをリアルタイム監視することはできない。そのため、電子メール配送ログから SMTP 認証に関するログの抽出工程だけは、電子メール送信サーバ上で行う。これは電子メール配送ログ全体を直接ログ収集用 NFS 領域へ保存した場合に生じる大学基幹ストレージの IO 負荷と、電子メール送信サーバ上ですべての処理を行った場合に生じるサーバ負荷とのバランスをとった結果である。

gusyslog はログ収集用 NSF 領域に逐次追記される SMTP 認証ログをリアルタイム監視し、統計・解析を行う。その結果、異常な状態であると判断された場合には、セキュリティ管理者へアラートメールを発信すると共に情報館事務室設置のネットワーク対応パトライトを点滅させることで異常を通知する。

5.1　導入効果

この仕組みの導入以前は、宛先不明で返送されるエラーメールの量などから担当者が経験を基に随時電子メール送信ログをチェックするなどしていた。しかし、判断基準が個人の感覚にゆだねられていたため、過去には不正アクセスの検知までに 20 日以上を要し 20 万通を超える不正メールの送信を許してしまう事例もみられた。こうした過去の事例に対してこの仕組みを適用した結果、すべての事例において不正メールの送信開始から 10 分以内、送信メール数が 100 件に届かない時点で検知・通報されることが確認できた。また、検知と同時にパトライトが点滅することで、システム担当でない職員が「なにか異常が起きている」ことを認識でき、管理者に注意を促すことで対応を早める効果もあった。

さらに、担当者が出張などで不在の際や、夜間や休日であっても不正アクセスの検知・通報が行われるようになり、担当者が定期的なログチェック業務から開放されただけでなく、迅速なインシデント対応が可能となっている。

6　事例２）解析結果の Web 可視化

学内上層部に対して次世代型ファイアウォールの導入効果を説明するための資料として、学外から学内へ接続を試みる通信コネクションのうち同機器により遮断された（不正アクセスと考えられる）コネクション数と、許可された（正規サービス宛の）コネクション数を日次でグラフ化し四半期に１度報告している。

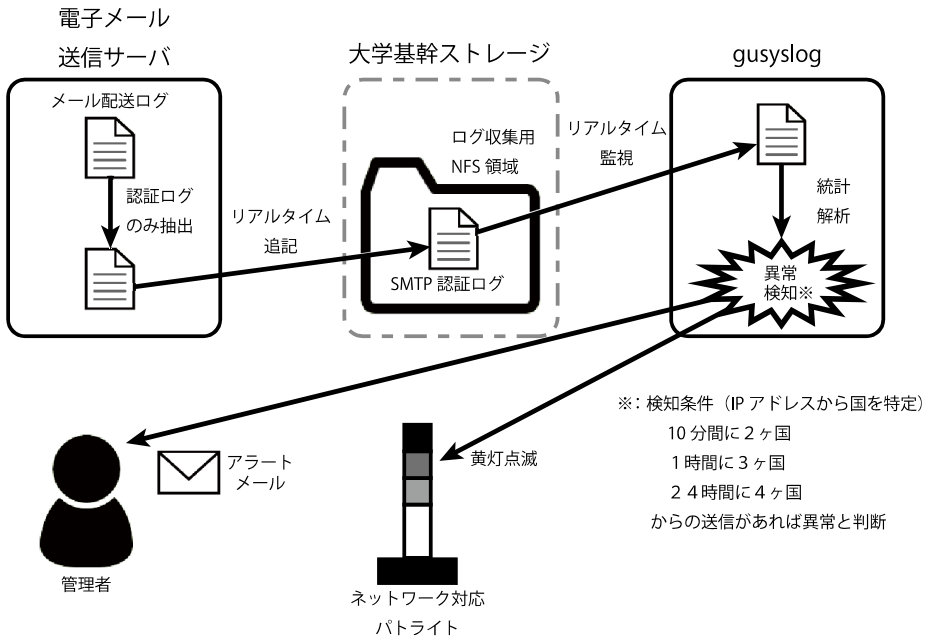

図３　事例１）リアルタイム異常検知構成図

通信コネクションの遮断/許可状況が記録される次世代型ファイアウォールのトラフィックログは、各種システムが出力するログファイル中最大のサイズを誇っており、連日 2 億行を超えるログが記録される。分量的に人手での集計作業が現実的ではないため当初より集計プログラムを用いた半自動化がなされていたが、１日分のログ集計には Ruby による実装では 90 分強、高速な Crystal を用いても 25 分程度が必要であり、グラフ作成の前準備として四半期分のデータ処理に 30 時間以上を要する状況があった。この問題に対応するため、岐阜大学情報連携統括本部では gunetlog を中心として図４に示す仕組みを構築した。

まず、日次のトラフィックログを集計し SQLite[10] DB へ格納するプログラムを夜間バッチとして実行する。このようにして前日までの集計結果を DB からいつでも取得可能とした上で、そこから Chart.js[11]用のプロットデータを自動生成することで過去１年間の集計結果推移を常時 Web ブラウザ上で閲覧可能とした。

ここで使用した Chart.js は JavaScript と HTML5 のキャンバス機能を利用し、ブラウザ上で各種グラフをインタラクティブに表示させるオープンソースライブラリである。今回は毎日更新されるプロットデータの定義部分を独立した JavaScript ファイルとして生成することとし、グラフのカラースタイルや軸設定など固定部分は表示用 HTML ファイル内に静的に記述した。プロットデータは gunetlog 上で生成された後、Web サーバの公開領域へ転送される形となっており、仮に不慮の事態が生じてデータ生成に失敗したとしても、直前までの情報は Web 上で確認できる。

6.1　導入効果

この仕組みにより、会議資料作成のために長時間のプログラムを実行してデータを集計し、さらに担当者が手作業でグラフ化していた作業が自動化された。また、これまで四半期に 1 度だけ可視化されていたインターネット上の不正活動の実態が Web ブラウザ上で常時確認可能となった。さらに、日次の集計結果は DB に保存されているため、任意の期間を違った視点で統計処理する際にデータの取り直しが不要となり、作業時間の短縮にも繋がっている。なお、この仕組みは本事例だけでなく、キャンパス情報ネットワークに接続している端末の利用状況可視化などにも利用されており、今後もその適用範囲が広がることが期待される。

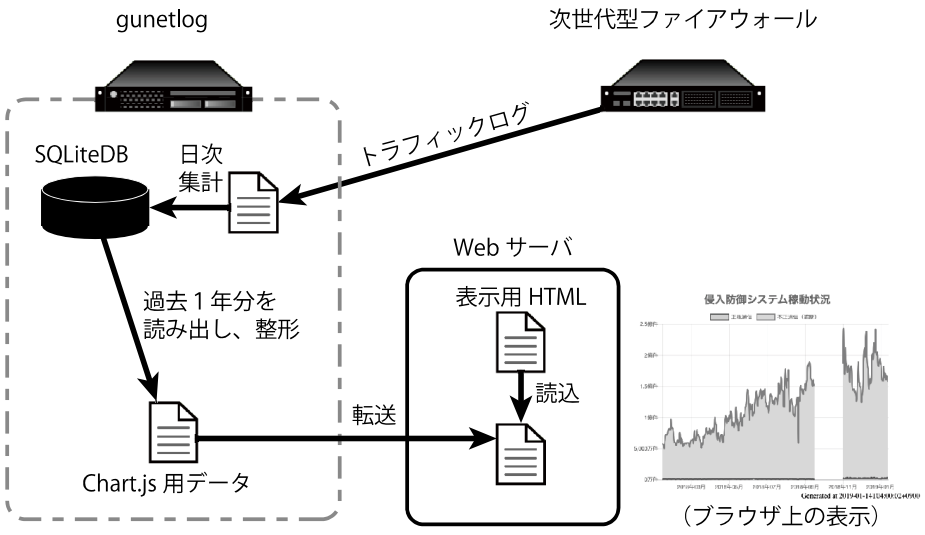

図４　事例２）解析結果の Web 可視化構成図

7　まとめ

ログ解析といった大量のデータを処理する業務においては、プログラムを利用した集計処理が不可欠である。処理内容をアルゴリズム化することにより、担当者のケアレスミスや判断のブレを防止し、毎回同じ基準で処理されたデータが取得可能になる。今回、岐阜大学情報連携統括本部ではそうした処理を定期、不定期に自動実行させ、アウトプットを担当者へプッシュする仕組みを構築した。このことにより、さらなる省力化が実現され、同時にインシデント対応の迅速化や作業漏れなどヒューマンエラーの回避が可能となった。

今後は既存プログラムのアルゴリズム検証を進め、技術系職員間でノウハウを共有して持続可能なシステムとして運用していく体制を目指す。また、こうした取り組みを利用停止期限が近づいた利用者へのリマインダ送付など、ログ解析業務以外にも展開していきたい。

参考文献

[1] “RFC5424: The Syslog Protocol”, https://tools.ietf.org/html/rfc5424, (参照 2019-02-14)

[2] “awk.1p”, http://man7.org/linux/man-pages/man1/awk.1p.html, (参照 2019-02-14)

[3] “grep.1p”, http://man7.org/linux/man-pages/man1/grep.1p.html, (参照 2019-02-14)

[4] “sort.1p”, http://man7.org/linux/man-pages/man1/sort.1p.html, (参照 2019-02-14)

[5] “uniq.1p”, http://man7.org/linux/man-pages/man1/uniq.1p.html, (参照 2019-02-14)

[6] “The Perl Programing Language”, https://www.perl.org,(参照 2019-02-14)

[7] “オブジェクト指向スクリプト言語 Ruby”, https://www.ruby-lang.org/ja/, (参照 2019-02-14)

[8] “Welcome to Python.org”, https://www.python.org/, (参照 2019-02-14)

[9] “The Crystal Programing Language”, https://crystal-lang.org/, (参照 2019-02-14)

[10] “SQLite Home Page”, https://www.sqlite.org/, (参照 2019-02-14)

[11] “Chart.js | Open source HTML5 Charts for your website”, https://www.chartjs.org/, (参照 2019-02-14)