Fluentd

寺井 大地，市川 燿
Daichi TERAI，Hikaru ICHIKAWA

1　はじめに

近年, システムの稼働状況によって仮想マシンの数が増大する仮想化やサービスを利用するクライアント端末のモバイル化に伴い web システムや企業の社内システムはますます大規模化している. このようなシステムはアクセスログやデータベースログ, システムログ等様々なログが大量に生み出す. 以前は情報セキュリティ対策や企業内コンピュータの不正利用を防止するためにログの収集を行っていた.

しかしながら, 現在ではこれらのシステムを運用するに当たって出力されるログを管理するだけでなく分析することによりサービスの向上がはかられている. ログの収集・管理においてその主目的はシステム内で何が起きているのかを把握することである. しかしながら, ログ収集・管理においてはログを出力するハードウェアやソフトウェアが多岐にわたっていることや, 多種多様なログが散在しており一元的に管理されていないことといった課題が残っている.

また, 大量のログが出力される状況下においては膨大なログの中から意味のある情報を抽出することは重要である. システムの管理者は大量のログを効率よく収集・管理し, 情報の抽出や分析を行い, 活用することを求められている. このような状況に対応するために Fluentd というログ収集・管理ツールに注目が集まっている. 本稿では Fluentd の概要と実用例, 及び Fluentd の展望について述べる.

2　Fluentd の概要

2.1　Fluentd とは

Fluentd は 2011 年 9 月に初リリースされ, トレジャーデータ株式会社の開発者およびコミュニティベースで開発が進められているログ収集・管理ツールである. プログラムは Ruby で書かれており, オープンソースソフトウェアである. 大規模な社内システムや web システムを取り扱っている企業向けに開発され,1 日に 5TB 以上のログを Fluentd で処理するユーザーもいる [1)] . 各クライアントにインストールされた Fluentd のエージェントがログの収集や加工, 分析処理などを行い, エージェント同士が連携してログの管理を行う.Fluentd の内部においてログの収集・管理は「インプット」「バッファ」「アウトプット」の 3 層に分けて行われる. 各層がプラグインを導入することのできるアーキテクチャになっており, 用途に応じてプラグインを追加・変更するだけで簡単に機能を追加・変更することができる.

2.2　プラグイン

Fluentd はプラグインを導入することでログの収集方法や保存先をユーザーが柔軟にカスタマイズすることができる.Fluentd のプラグインはインプットプラグイン, バッファープラグイン, アウトプットプラグインの 3 種類に分類される. インプットプラグインはログの出力元からログを検索, 取得するプラグインである. データを入力元から収集する処理, 収集したデータを Fluentd の内部で扱える JSON 形式に変換する処理, およびタグや時刻を付与して Fluentd の内部で扱うイベントを発行する処理の 3 つの処理を行う.

バッファープラグインはインプットプラグインから入力された情報をアウトプットプラグインでターゲットに対して出力するまでの間のデータ管理を行うプラグインである. データはキューとチャンクという塊で一時保存される.

アウトプットプラグインは収集されたデータを用途に応じて出力するプラグインである. インプットプラグインから送られたログのデータのフォーマットを変換する処理とアウトプット先への出力をする処理の 2 つの処理を行う.

Fluentd は入力と出力の処理をプラグインとして追加・変更するだけで運用しているシステムの環境に応じたログ収集・管理ツールとなる. Fig.1 に Fluentd 内部における処理の流れを示す.

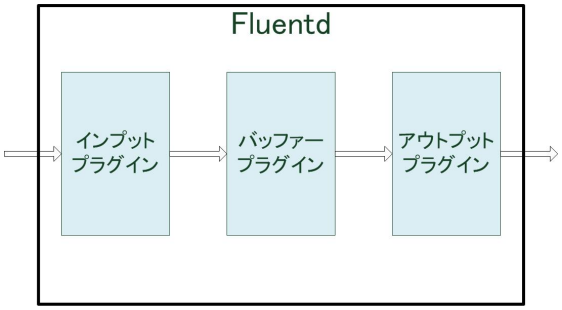

Fig.1　Fluentd 内部の処理の流れ

2.3　Fluentd の利点

Fluentd を導入していないシステムではそれを運用するに当たり, システムにおける正常性のチェックやアクセスログからユーザーの行動分析など多くの目的を実現するためにログを出力し, 蓄積する. 収集するログの種類によって処理の方法が決められており, 実行するスクリプトによって決められた処理を行う. そのため最大でログの種類と同数のスクリプトが必要となる.Fluentd を使用

すると個別に用意していたスクリプトをすべて Fluentd に置き換えることができるため, 構成がシンプルになる. Fig.2 に従来のログ収集・管理例,Fig.3 に Fluentd を用いた場合のログ収集・管理例を示す.

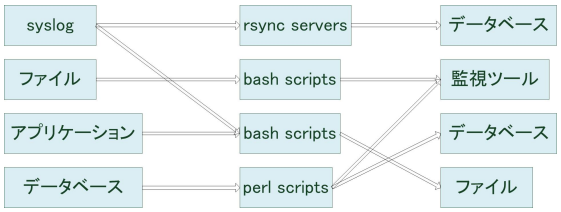

Fig.2　従来のログ収集・管理例

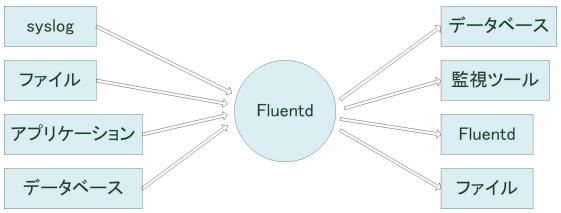

Fig.3　Fluentd によるログ収集・管理例

また,Fluentd を導入していない従来のシステムでは 1 日に 1 回夜間にバッチ収集をしてデータストアへ格納したり, 分析するためのスクリプトを実行したりすることが多くリアルタイムに状況を把握することが困難であった. リアルタイムに分析する場合はその度にスクリプトを実行しなければならなかったが,Fluentd を導入したシステムは対応したプラグインを入れるだけでリアルタイムなストリーミング処理が可能となる.

信頼性の面においても Fluentd は優れている. ログの保管先にログデータを送信している時, 送信先のサーバがダウンするといった予期しない障害が発生する場合がある. その場合ログデータの一部が失われてしまうことが考えられる. しかしながら,Fluentd にはバッファリングというログデータを一時保存する機能が備えられている. そのためデータの損失を軽減することができる. 出力元からインプットされたデータは Fluentd 内部に一度チャンクという固まりとして保持される. このチャンクはデータの出力先に書き込まれたことを確認してから消去されるため, 障害によるデータの損失を押さえることができる.

また,Fluentd ではユーザーが独自に必要な機能を自由に追加することができる. ユーザーが作成したプラグインは web 上に公開することができ, 現在では 200 以上のプラグインが公開されている. 収集されたログは一度 JSON という形式へと変換される.Fluentd においてこの形式はタグ, 日時, レコードの 3 要素で構成される. 構造化されたデータを扱うためパースが容易でコンピュータにとって可読性が高い.

3　Fluentd の導入事例

3.1　COOKPAD での導入事例

クックパッド株式会社ではユーザーのアクセスログをテキストファイルに書き出して夜間にバッチ収集を実施し,MySQL に格納して管理を行っていた. しかしながら,COOKPAD には 1500 万のユニークユーザーが存在し, サイト上には 170 万のレシピがあるため, 収集するログの量は膨大である. 収集したログはストレージにシリアライズして保管することしか行っていなかった. そのためサービスの開発やサービス品質の向上に利用出来ていないという問題点があった. この問題を解決するために Fluentd を導入した. その結果, ログの統計データを出力しユーザーの行動傾向分析することができるようになった. また, 夜間に行っていたバッチの収集処理についてもリアルタイムに行うことが可能となった.

3.2　すれ違い通信中継所での導入事例

NINTENDO は 2013 年 8 月にすれ違い通信中継所というサービスを提供し始めた. このサービスは世界中にある 10 万ものアクセスポイントに,NINTENDO3DS を近づけると自動で情報を蓄積し他のユーザーとデータの交換ができるというサービスである. 各アクセスポイントに Fluentd を導入することでユーザー同士のコミュニケーションをより活性化させるとともに, どのようなユーザーがどのくらいの人数, どのアクセスポイントですれ違っているか 1 分程度の極めて短いタイムラグで把握することが可能である [2)] .

4　Fluentd の展望

現在 Fluentd が対応している環境は UNIX 系の OS のみである. しかしながら, 世界中で最も使用されている OS は Windows である. 今後は Windows 環境においても動作するバージョンの開発が期待される.Fluentd はユーザーによって機能の拡張ができるという点でオープンソースのメリットを生かした開発成功例といえる. Fluentd はまだまだ発展途上のソフトウェアで今もなおバージョンアップが繰り返されており, プラグインも次々と公開されている. 今後もユーザーによるプラグインの開発が進み,Fluentd でログ収集・管理の可能な範囲が広がっていくことが期待される.

参考文献

1) Overview of fluentd — fluentd.
http://docs.fluentd.org/articles/architecture.
2) ニンテンドー 3ds ｜社長が訊く「すれちがい通信中継所」｜ nintendo.
http://www.nintendo.co.jp/3ds/interview/streetpass_relay/vol1/index4.html.