SPring-8 加速器制御系における診断ログ収集システム

DIAGNOSTIC LOG COLLECTING SYSTEM FOR ACCELERATOR CONTROL AT SPRING-8

藤原綾潜#,A), 松本崇博 A), 山鹿光裕 A)
Ryosen Fujihara#,A), Takahiro Matsumoto A), Mitsuhiro Yamaga A)
A) Japan Synchrotron Radiation Research Institute

Abstract

In this paper, we report on a diagnostic log collecting system for accelerator control system at SPring-8. The SPring-8 accelerator control system is based on distributed computers using approximately 360 hosts. Diagnostic logs (e.g. messages between control terminals and equipments, records of the data collection) from each host are collected and transferred into a single log server, and are investigated once a trouble occurs. However the amount of logs transferred to the log server reach the size of 30 GByte per day, which is equivalent of 30 billion rows. It is very time-consuming to utilize these logs because of its large volume and varieties in the data format. To improve the efficiency, a log analysis system using fluentd and kibana is constructed and for the moment applied to monitor computer resources and web logs. We found that the log analysis system is efficient to find the causes of the trouble immediately.

1. はじめに

SPring-8 加速器制御系の計算機は、機器のデータ収集を行った記録や、制御端末と機器との間のメッセージングの記録を、プログラムログとして出力する。ログの量は約 360 のホストを合計して、一日におよそ 350GByte、行数にして 30 億行（毎秒 3 万 5 千行）が書き込まれる。SPring-8 では平成 24 年に、この大量のログを一つのサーバに集約し、閲覧する環境を整えた。集約されたログはトラブル発生時の原因解明に使用される。機器担当者はログ閲覧サーバにログインし、ターミナル上でコマンドや専用のスクリプトを使用してログ解析を行うが、ログサイズが大きい場合や、複数のログを追跡する場合は多くの時間と労力を消費する。この課題を解決するため、ログデータの抽出と可視化の手法を調査し、ログ解析ソフトウェアの fluentd[1] と可視化ツール kibana[2]を用いたログ解析環境を構築した。

本報では、ログ収集システムの現状について触れたうえで、ログデータの抽出と可視化の概要を述べ、実施例を示す。

2. 診断ログの収集

2.1 ログの出力

加速器制御系の計算機が出力するログは、計算機上の syslog デーモンによってログサーバに転送される。主要なログの例として、冒頭で述べたメッセージングプログラムのログを Figure 1 に示す。ログにはメッセージの方向と送信先/受信元の情報、メッセージを識別するための ID と、メッセージの内容が含まれる。

```
Jul 25 11:15:41 [10511] msg::recv@opcon01
msgID=15856_safrdygui_oper_opcon01@7381437.1
Jul 25 11:15:41 [10511] msg::send@opcon01
msgID=15856_safrdygui_oper_opcon01@7381437.2,
svoc=15856_safrdygui_oper_opcon01/get/safety.../status@15856
...
```

Figure 1: Log sample from messaging program.

これら制御プログラムのログが全体の大部分を占めるが、それ以外のものとして OS のシステムログや一般的なアプリケーションのログもある。加速器制御系データベースのデータ閲覧用 web サーバは、自動更新によるアクセスも含めると一日で 70 万行（毎秒 8 行）程度の http アクセスログを出力する。

2.2 ログの保存

計算機から送信されたログは、ログサーバに保存される。ログサーバがログを蓄積しているパーティションは NFS サーバとして export しており、ログ閲覧サーバがこれをマウントする(Figure 2)。

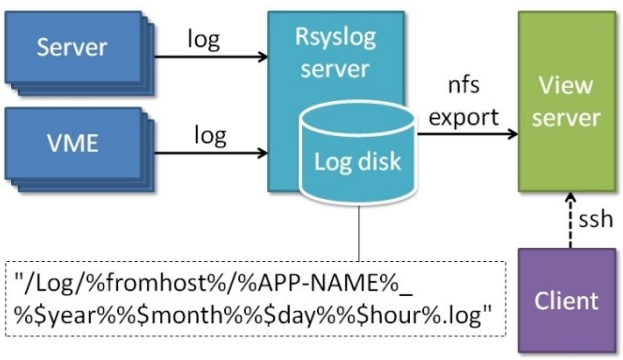


Figure 2: Diagram log collecting system.

#fujihara@spring8.or.jp

Figure 2 中に示す通りログは機器のホスト名(%fromhost%)ごとにフォルダ分けされ、プログラム名(%APP-NAME%)と日時を含むファイル名を付け、1時間を1ファイルとして保存する。ログのパスにホスト名を含めると、ログサーバがログを書き込む際に頻繁に名前解決を行うため、ローカルに DNS キャッシュを持たせ、DNS サーバとの通信を減らしている。

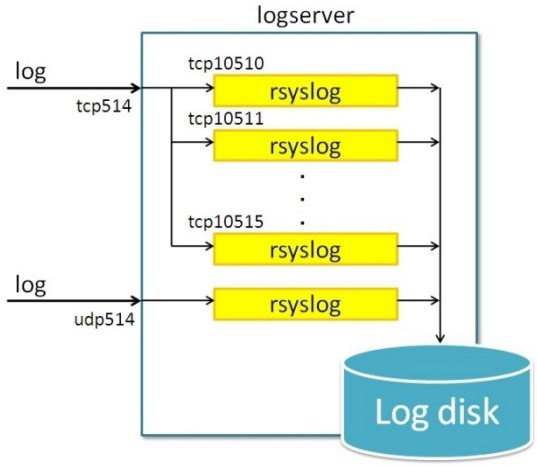


Figure 3: Load balancing using multiple rsyslog task.

ログサーバは、転送されたログを tcp514 ポートで受け付け、ログサーバの内部処理によりポート10510～10515 にランダムに振り分ける(Figure 3)。各ポートにはそれぞれ rsyslog タスクが待機しており、転送されたログを共通のローカルディスクに保存する。rsyslog のタスクはそれぞれ 1 つの CPU コアをリソースとして使用しており、ログの書き込み処理を複数タスクに振り分けることで負荷分散を行っている。

保存したログはローテーションにより 20 日間保持した後に削除する。ログの圧縮による CPU 負荷の発生を抑えるため、ログの圧縮は行わず、代わりに十分な disk 容量を用意してテキストのまま保存している。本報執筆時の現在、rsyslog タスクが使用している CPU コアの使用率は平均 20%程度、また 11.0TByte の disk 容量に対し 6.5TByte のログが保存されている。メモリの使用は見られない。ログサーバのハードウェア仕様を Table 1 に示す。

2.3 ログの利用と課題

機器担当者がログを見る際はログ閲覧サーバに ssh ログインして直接ログテキストを開く。

ログはほとんどの場合、アラーム発生時や不具合が報告された時の原因解明に使用される。先述のメッセージログであれば、機器アラームが発生した際、メッセージがどこで途切れているかを追跡することで、問題箇所のホスト名や機器名を特定できる。データ閲覧用 web サーバのログの場合、特定のホストからの連続アクセスや cgi スクリプトのエラーをログの中から探す。

ログの調査は grep による絞り込みで行う。トラブル発生時間が分かっている場合はその付近の時間帯に絞り込む。ホスト名やエラーメッセージが明らかな場合、それらをキーワードとしてさらに絞り込む。調査対象の範囲を絞ったうえで、最後は担当者が目でログを追って原因を特定する。

grep によるログ調査が困難な場合もある。ログサーバに 1 時間ごとに保存されるログファイルは大きいもので 100MByte を超え、行数も相応のものとなり、絞り込みの後でも行数が多く残ると調査を行うことができない。また調査対象は 1 ファイルに限らず、メッセージングのログの場合はホストからホストへの追跡が必要で、調査対象も複数のファイルに拡がり、多くの時間と労力を消費する。

特定のログに限れば、スクリプトを用意することでログ調査の効率を上げることができる。メッセージログに関しては、解析用スクリプト chklog を用意した。これは、ホスト名や時間範囲などの条件を引数として入力することで、全メッセージログの中から条件に合った行を選び出すものである。一方、全てのログに適用可能なスクリプトを制作するのは難しい。これは、ログの書式がログの種類ごとに大きく異なり、日時の表記一つをとっても統一されておらず様々だからである。

この課題を解決するものとして、以降に述べるログの JSON 化や可視化の手法が挙げられる。

Table 1: Server Specifications

CPU	Xeon E5620 2.4GHz 4core ×2CPU
MEM	4GB
Disk	2TB×8disk (RAID6)

3. ログ分析と可視化

3.1 ログの JSON 書式化

ログ解析が困難な理由として、テキストであるために文字列処理が必要なこと、ログの種類ごとに書式が異なることが挙げられる。

これを解決するため、ログ解析ソフトウェア fluentd を導入し、ログ情報をキーとバリューの形で JSON 書式にしたうえでデータベースへの保存を行った。JSON 化によりテキスト処理の必要が無くなり、またデータベースへの保存によりログの種類に関わらず一定の方法でアクセスできるようになった。データベースを用いたログ収集システムの模式図を Figure 4 に示す。

ログ解析サーバはログフォルダをリードオンリーで NFS マウントし、ログにアクセスする。ログサーバにはログ解析ソフトウェア fluentd と各種プラグイ

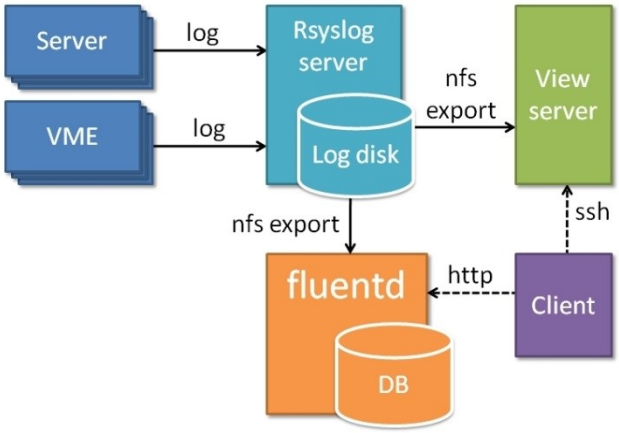


Figure 4: Diagram log collection system plus fluentd.

ンがインストールされており、テキストログの JSON 化とデータベースへの保存を行う。fluentd はログの取り込み、加工、出力に関して豊富なプラグインを持ち、プラグインの機能はコンフィグファイルへの短い記述のみで適用できる。

ログを JSON 化する際は、テキストログの取り込みを行った後、parser プラグインによりログテキスト内から必要な情報をキーとバリューの形で取り出す（パース）。パースの単純な例を Figure 5 に示す。

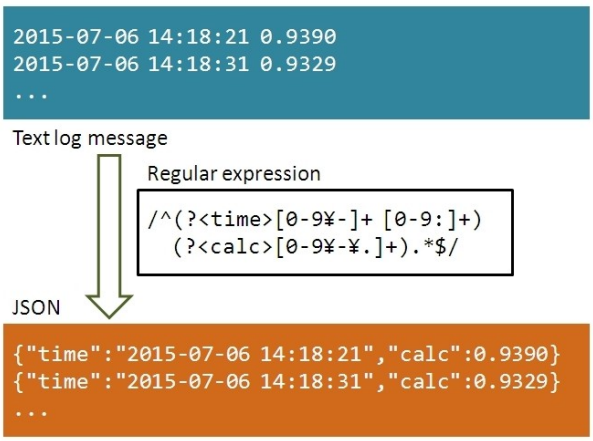


Figure 5: Parsing strings and values in log messages using fluentd plugin.

Figure 5 では日時と計算値をスペースで分けて記したログを処理している。図中に示すようなパース用書式フォーマットを用意し、書式にマッチした箇所のテキストを取り出してキーを与え、JSON 形式のデータにすることができる。この方法により、書式が一定でありさえすれば、どんな書式のログでもデータとして取り出すことができる。

取り出したログデータは、fluentd に用意された各種データベースへの出力プラグインによりデータベースへ保存する。データベースは用途に応じて mongoDB[3]又は elasticsearch[4]を用いた。

3.2 ログデータの可視化

データベースに保存されたログデータは、cgi を作成して web ブラウザ上でテーブルやグラフ表示を行うことで可視化が可能であるが、ログの種類や可視化の項目ごとに毎回異なる cgi を作成する必要がある。

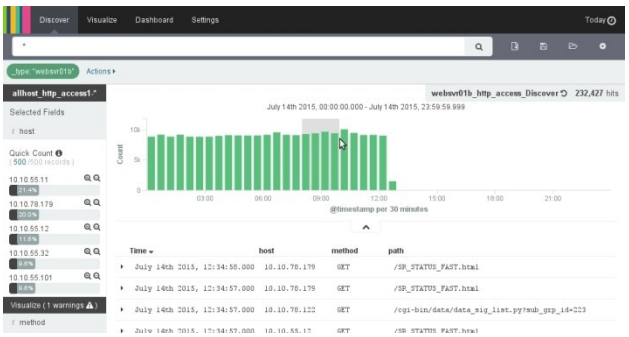
現在は可視化ツール kibana への移行を進めている。kibana はデータの絞り込みやグラフ表示のバリエーションに優れており、cgi を作成することなく、ログデータの多様な可視化が可能となる。kibana の機能については実施例で触れる。

4. 診断ログの可視化例

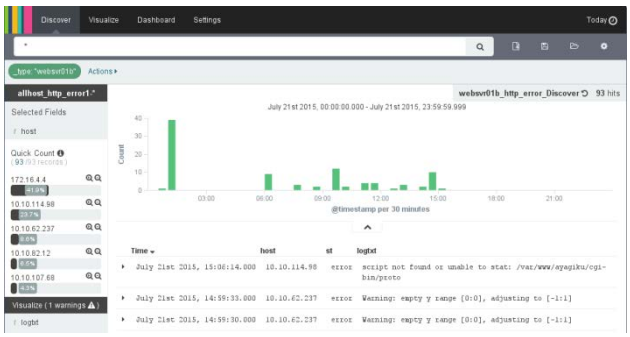
4.1 データ閲覧用 web ログの可視化

2.1 でも触れた、加速器制御系データベースデータ閲覧用 web サーバのログ可視化例を Figure 6 に示す。

ログは一般的な apache 形式である。必要なログデータとして、時間、リモート IP を取得し、アクセスログについてはアクセス方法(GET/PUT)とアクセスパス、エラーログについてはステータスとエラーメッセージを取得した。アクセスログについては fluentd に用意されている apache2 フォーマットを使用してパースを行い、エラーログについては正規表現を用意した。



(a) access_log



(b) error_log

Figure 6: Visualization of http log (screen capture of kibana 4).

Figure 6 中、(a), (b)がそれぞれアクセスログ、エラーログを示す。いずれも画面上部はログカウントのヒストグラムを示し、ログ件数の変化を知ることができる。下部はパースにより得られたログデータが、上部のヒストグラムと同じ時間範囲でテーブル表示される。

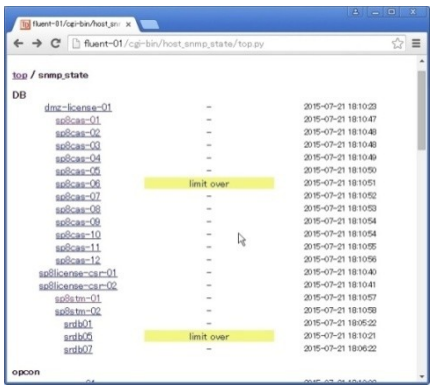
ブラウザ上のマウス操作により、時間範囲の絞り込み、ワード検索、特定のキーに対する簡易的なデータ分布の表示ができる。これら複数の絞り込み機能を利用することで、直接ログファイルを grep コマンドで調べる場合に比べて、短時間で効率良くトラブルの原因を解明できる。

4.2 snmp ログを利用した計算機リソース監視

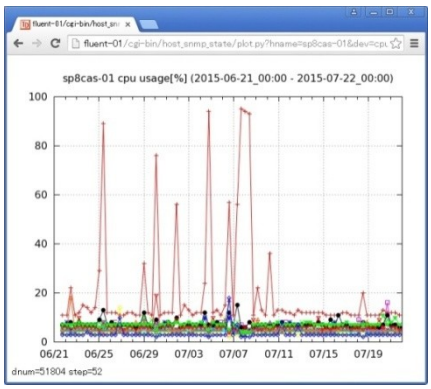
計算機の簡易的な監視を行うため、計算機監視サーバ上に snmp コマンドを発行するスクリプトを置き、監視下の 87 台の計算機の死活およびハードウェアの使用状況を一定時間毎にログとして出力している。


```
{"time":"2015-07-21 16:10:59","host":"sp8stm-01","info":"0",
"cpu":{"0":{"us":"44","st":"10"},"1":{"us":"15","st":"10"}},
"mem":{"0":{"us":"4.0","st":"10"}},
"hdd":{"/":{"us":"9.8","st":"10"},...,"/home/sr":{"us":"88.2","st":"10"}}}
```


Figure 7: Log of snmp for monitoring servers.



(a) Overview of alarm for each host



(b) Graph of hardware load

Figure : 8 Computer monitoring screen using the snmp log.

ログは 1 ホストに対する 1 回のチェック結果を一行の JSON 形式のテキストで出力する。ログの例を Figure 7 に示す。計算機の代表的な状態と CPU、メモリ、disk の使用率が含まれる。ログのパースには fluentd に用意されている json フォーマットを用い、ログ内のすべての数値をログデータとして取得した。本例ではデータを mongoDB に保存し、専用の cgi を用意して可視化を行った。ブラウザで表示させた計算機監視のトップ画面を Figure 8(a)に示す。

画面には死活やリミットオーバーなど各ホストの代表的な状態を表示した。画面内のホスト名をクリックするとホストごとの詳細が示され、ハードウェアの使用率の変化をグラフ表示できる。CPU 負荷のグラフの例を Figure 8(b)に示す。この監視は多種類の計算機や OS が混在する環境で、ホストごとに監視エージェントをインストールすることが困難な場合に使用している。

5. まとめと課題

加速器制御系のログ収集・閲覧環境を整えた。さらに、サイズの大きいログを調査する際に発生する問題点を改善すべく、ログデータの抽出と可視化の環境を整え、実施例を示した。

現在ログ解析サーバは仮想マシン上で試験的に運用中であり、今後大量のログを処理するため、物理サーバの導入と負荷試験を行う予定である。

また、SPring-8 制御系は現在、制御フレームワークを従来の MADOCA から MADOCA II[5]へ移行する過渡期であり、これに伴ってより複雑なログ解析の需要が見込まれる。本報ではログのパースによる単純な JSON 化処理のみ示したが、今後は演算処理など多様なログ解析ができる環境を整備したい。

参考文献

[1] http://www.fluentd.org/
[2] https://www.elastic.co/products/kibana
[3] https://www.mongodb.org/
[4] https://www.elastic.co/
[5] T. Matsumoto et al., “MADOCA II 制御フレームワークの SPring-8 への実装及び機能拡張”, Proceedings of the 11th Annual Meeting of Particle Accelerator Society of Japan, Aomori, Aug 9-11, 2014.