研究論文

鳴門教育大学情報教育ジャーナル No.18 pp.65-68 2021

情報基盤センター業務における時系列データベースの活用

曽根直人*

情報基盤センターは，鳴門教育大学における学内 LAN や端末室などの共有 ICT 環境の整備・運用を行なっている。安定した ICT サービスを提供するためには，機器の利用状況やログ情報を確認し，機器やサービスの状況を把握する必要がある。しかし，それらの情報は本学のような単科大学の規模でも膨大なデータとなり，そのままの状態では活用が難しい。そこで zabbix のような監視ツールにより，機器の情報を SNMP により取得し，グラフによる可視化やしきい値を用いて利用状況を把握することが行なわれている。監視ツールによる機器やサービスの状況把握は非常に便利であるが，情報基盤センターの運用ではさまざまな情報について，より柔軟に可視化やログの検索を行いたい要望がある。そこで，より柔軟な可視化を目指し，時系列データベースを活用したシステム状況把握用ダッシュボードを開発した。

［**キーワード**：時系列データベース，可視化，ダッシュボード，ログ管理］

1. はじめに

サーバプログラムから出力されるログ情報やセンサから取得した値などは，時刻情報と密接に結びついた時系列情報である。従来の RDB でも時間を扱うためのデータ型を持っており，テーブルに時刻情報を記録し扱うことは可能であるが，近年，時系列情報の蓄積や検索をより柔軟に行なえる時系列データベースが開発され，注目を集めている。

情報基盤センターのシステム運用で参照する情報も，時系列情報を多く扱っており，以前より OSS の munin[1]と自作のプラグインを使って可視化を行なってきた[2]。しかし，これらのシステムは汎用性に乏しく拡張性も困難であった。そこで時系列データを柔軟に扱うことができる時系列データベースを利用し，情報システムの運用に必要な情報の蓄積と可視化を行なった。可視化用のソフトウェアを利用し，データベースに対してクエリを実行すれば可視化が行なえる柔軟なダッシュボードを構築した。

2. 時系列データベース

MySQL や PostgreSQL といった RDB は，事前にデータ型を組み合せてテーブルを定義し，そこへデータを格納するスキーマデータベース(DB)である。データ型に日付型や時刻型を持っており，時系列データも扱うことはできるが，テーブル作成時に決めた構造で保存する必要があり，柔軟性に欠ける。最近は，スキーマレスでデータを柔軟に保存できるスキーマレス(NoSQL)DB も開発，利用されており，MongoDB などのソフトウェアが有名である。さらに，IoT により大量に生成されるセンサ情報やログ情報といった時系列でのデータを効率良く扱うために，DB の主キーとして時刻情報を利用することにより，時系列情報の保存に特化した時系列 DB の開発，利用も進んでいる。時系列 DB には RDB(PostgreSQL)を拡張した TimescaleDB[3] や NoSQL の技術を利用した InfluxDB[4]などがある。

時系列 DB では，一定期間におけるデータのサンプリングや合計，平均，分散を求める時系列分析関数がサポートされており，クエリ言語で問合せを実行するだけで，分析が実行できる。

今回開発したシステムでは時系列 DB として，InfluxDB(OSS 版）を採用した。InfluxDB は influxdata 社が開発する有償ソフトウェアであるが，商用版からいくつかの機能を省いた OSS 版が github 上で公開されている。また Debian や Ubuntu，RedHat など主要 OS のバイナリパッケージも提供されているため，これらの OS では簡単に導入することができる。さらに Telegraf という機器やセンサの情報を収集するためのエージェントソフトウェアや influxDB 1 系と組み合わせてデータの可視化を行なう Chronograf，リアルタイムデータ処理を行なう Kapacitor が提供されており時系列データの収集，保存，可視化，監視に必要なソフトウェアが提供されている。これらソフトウェアを総称して TICK Stack(図 1)と呼んでいる。

* 鳴門教育大学大学院 高度学校教育実践専攻 自然・生活系教科実践高度化コース(技術・工業・情報科教育実践分野)

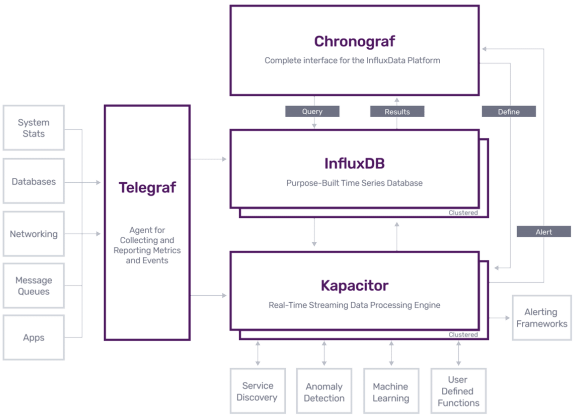

図 1 TICK Stack 概念図
（influxdata 社 Web より引用[4]）

3. 時系列 DB の活用

情報基盤センターの各種システムの値やログを蓄積するために利用したシステムは，表 1 の環境で稼動している。

3.1. moodle ログオン

OSS の学習支援システム moodle を稼動しているホストでは，Telegraf をインストールし，ホストの負荷を記録している。さらに Telegraf から shell script を実行し，以下のような SQL を発行することで moodle へのログオンユーザ数を取得している。

```
use moodle;
select count(username) as active_users
  from mdl_user AS u
  WHERE u.id > 1
  AND
  lastaccess > UNIX_TIMESTAMP() - 60
```

この処理を 1 分間隔で実行し，1 分間隔毎のログオンユーザ数を把握している。この情報により，時間や曜日による moodle の利用状況について把握することができるようになった。(図 2)

3.2. HTTP Proxy のログ

情報基盤センターでは squid-3 を HTTP Proxy として利用している。第 7 期コンピュータシステムからは負荷分散装置が利用されており，Web ブラウザからのリクエストは 3 台の squid へ分散されている。そのため，Proxy のログを確認するには 3 台のサーバへ SSH 接続して確認するといった煩雑な作業が必要であった。昨今では，フィッシングサイトやマルウェアの C&C サイトへの通信をログから確認するような機会も多くなっており，作業の効率化が必要となっていた。

そこで，3 台の squid サーバでは，squid の access ログを rsyslog へ出力し，rsyslog 経由で syslog サーバへ集約するようにした。さらに syslog サーバでは，集約された squid の access ログを telegraf により influxdb へと格納した。squid が syslog へ出力するフォーマットは以下のように定義した。

表 1 ハードウェア，ソフトウェア環境

CPU	Intel Xeon L5520 2.27GHz
RAM	18GB
HDD	SCSI 146GB
OS	Debian 9.13
influxdb	1.8.5
Telegraf	1.18.1
Chronograf	1.8.5
Kapacitor	1.5.6

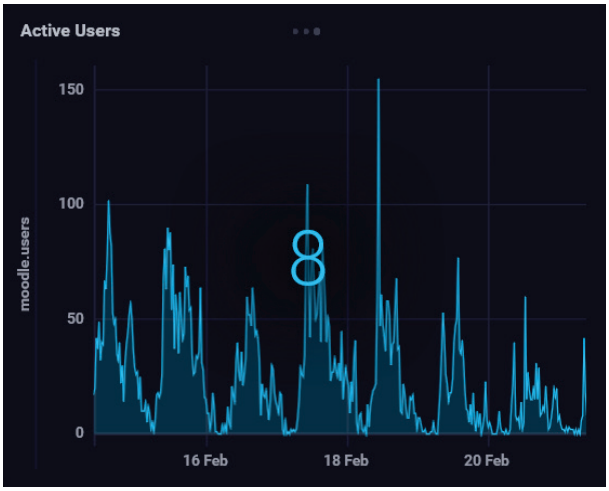

図 2 moodle ログオン数

```
  logformat  telegraf  %ts.%03tu  %>a  %<a  %<A
"%rm %ru HTTP/%rv" %>Hs %<st %<pt %mt "
  %{User-Agent}>h"
```

これにより，influxdb の格納に必要な粒度の時刻情報を含むフォーマットで syslog へ送っている。

syslog サーバでは Telegraf の logparser を用いて集約した access ログを influxdb へ格納する。

```
  [[inputs.logparser]]
    files = ["/var/log/proxy.log"]
    from_beginning = false
    name_override = "proxy"
    [inputs.logparser.grok]
      patterns = ["%{SQUID_LOG}"]
      custom_patterns = '''
  SQUID_LOG %{HOST:proxy:tag} %{NOTSPACE:progra
m:tag}                    %{NUMBER:timestamp:ts-
epoch} %{IP:client} %{IP:peer} %{HOST:peer_host}
"%{DATA:request}" %{NUMBER:status:int} %{NUMBER
:byte:int}     %{DATA:response}     %{DATA:type}
"%{DATA:ua}"
      '''
```

格納した情報を Chronograf で処理し HTTP Proxy のダッシュボードを作成した。図 3 はダッシュボードに表示している squid へのリクエスト数をグラフにしたものである。

ダッシュボードは，リクエスト数，バイト数のグラフおよびログ情報のテキストをテーブルに表示したパーツで構成した。

図 3 を描くためには Chronograf で以下のクエリを実行し，Visualization で Type として Line を選択している。

```
SELECT count("status") AS "proxy01" FROM "telegraf".."proxy" WHERE time > :dashboardTime: AND time < :upperDashboardTime: AND "proxy"='proxy01' GROUP BY time(:interval:) FILL(null)
```

influxdb へ格納した squid の access ログは可視化以外にも特定の URL へ接続した端末を調べるためにも利用している。この調査はダッシュボードではなく，サーバへ SSH 接続し，influx コマンドで直接クエリを実行している。例えば以下のクエリを実行すれば，過去 24 時間以内に squid 経由で特定の IP への通信を行なった情報を検索する。

```
select * from proxy where "peer"='通信先 IP' and time >= now()-24h tz('Asia/Tokyo')
```

3.3. プリンターの印刷枚数

情報基盤センターで運用しているネットワークプリンターは，オンデマンド方式になっておりユーザはネットワークプリンターへ印刷指示を出した後，実際に印刷するプリンターのところへ行き，隣のオンデマンド端末から ID/PW を入力した後，印刷を指示するようになっている。プリンターは学内の端末室に分散しており，プリント用紙の補充は情報基盤センターのスタッフが行なっている。補充を効率良くするために印刷枚数が把握できることが望ましいため，ダッシュボードで確認できるようにした。

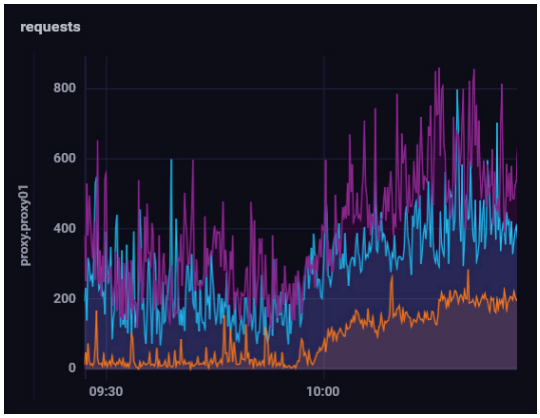

図 3 squid へのリクエスト数

Telegraf には SNMP により値を取得する機能が用意されており，次のような設定を行なうことでプリンターの印刷カウンターを取得するようにした。

```
[[inputs.snmp]]
  agents = [ "IP1", "IP2" ]
  community = "コミュニティ名"
  ## measurement name
  name = "snmp.printer"
  [[inputs.snmp.field]]
    name = "prtMarkerLifeCount"
    oid="Printer-MIB::prtMarkerLifeCount.1.1"
    conversion = "int"
```

図 4 に作成した Chronograf で印刷カウンターのダッシュボードを示す。印刷カウンターの値をそのまま利用したのでは変化が把握し難いため，influxdb の時系列分析関数 difference()を以下の用いて時間毎の差分値に変換しグラフを描いている。

```
SELECT difference("prtMarkerLifeCount") AS "diff_prtMarkerLifeCount" FROM "telegraf".."snmp.printer" WHERE time > :dashboardTime: AND "agent_host"='IP'
```

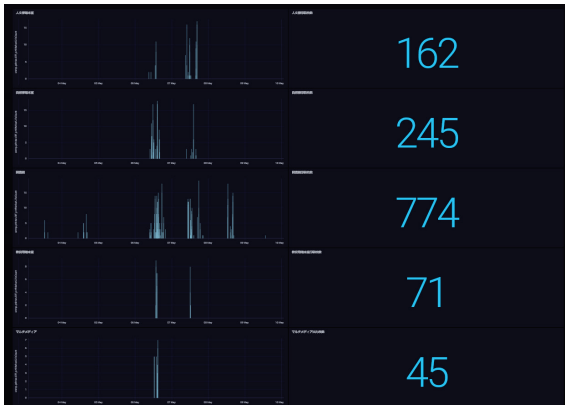

図 4 プリンター印刷枚数

3.4. 端末位置情報

proxy のログ情報から端末の IP アドレスは判明するが，その IP アドレスがその時間，学内のどの部屋から接続されていたかが判らなければ，ログからフィッシングサイトへのアクセスを検知したとしても，該当の通信を行なった利用者へ通知を行うことは難しい。そこで，学内の端末に割り当てた IP アドレスがどの部屋で接続されたのかを把握するために時系列 DB へスイッチの MAC アドレス情報を蓄積するシステムを実装した。図 5 にシステムの概要を示す。

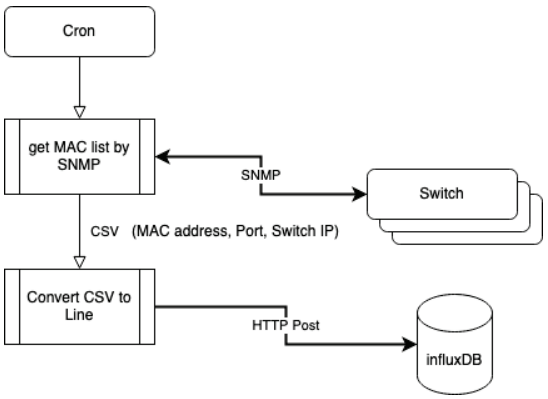

図 5 端末位置情報把握システム概念図

cron により定期的に ruby script を実行し，学内 LAN のスイッチに対して，SNMP により dot1dTpFdbAddress の情報を取得する。これにより，スイッチのポートとそのポートに対応する MAC アドレスの一覧を得ることができる。スイッチの uplink ポートでは多数の MAC が対応するため，script ではスイッチ毎に無視するポートの情報も扱えるようにしており，uplink の情報は SNMP により取得した一覧から削除する。そして以下の CSV 形式で取得した MAC 一覧を出力する

```
mac=xx:xx:xx:xx:xx:xx, port=yy, switch=IP
```

さらにこの CSV 形式のデータを InfluxDB line protocol へ変換し，HTTP Post を使って influxDB へ格納している。時刻情報が含まれていないが，これは influxDB へ Post した時刻が補われるためである。

実際に保存された情報を以下に示す。

```
> select * from fdb limit 5;
name: fdb
time                location        mac               port
1613347253000000000 192.168.191.100 00:24:a8:xx:xx:xx 42
1613347253000000000 192.168.191.100 00:24:a8:xx:xx:xx 41
1613347253000000000 192.168.191.100 00:24:a8:xx:xx:xx 40
```

各スイッチに対して 10 分毎に MAC アドレスの情報を取得し，DB へ格納している。この情報を利用することで，proxy のログ情報からセキュリティ上注意すべき通信が発見された場合，発信元 IP を検索すれば，その時間に発信元 IP の端末が学内のどこのスイッチの，どのポートに接続されていたのかを追跡できる。ポートまで判明すれば，どの部屋から接続されたのかも分かるため，実際に通信していた利用者を発見し，注意喚起を行うのに役立つ。

3.5. その他

情報基盤センターの influxDB にはこれら以外にも CO2 の濃度や稼働 PC 数，Wi-Fi AP ごとの接続クライアント数など多くの情報を influxDB へ格納してダッシュボードを作成し簡単に状況の確認ができるようにしている。これらの詳細については記述しないが，ダッシュボードへ表示したい数値は，それを取得する手順をスクリプト化し，得た値を Line protocol へ変換，influxDB へ格納する処理を cron などで定期的に実行すればよい。

4. まとめ

時系列 DB として influxDB を運用し，情報基盤センターのシステム運営で把握すべき様々な値を蓄積，可視化，監視することができた。TICK Stack の柔軟性により，従来利用していた munin や zabbix での監視と比べて容易に情報を追加，蓄積，可視化，監視できることがわかった。

ログ情報やセンサの出力など，処理すべき時系列データは増加している。TICK Stack など時系列データを柔軟に対応できるソフトウェアの登場により，それぞれのニーズに応じた時系列データの活用が可能となっている。

小・中学校の学習指導要領では，「データの活用」が含まれている。TICK Stack を利用すれば，身近なデータを簡単に蓄積，可視化が可能になるため，学校教育においての活用も可能性があると考えており，今後は情報基盤センターの運営以外の分野での活用を視野にした研究・開発を進めていきたい。

参考文献

[1] Munin, https://munin-monitoring.org（最終アクセス日：2021 年 1 月 7 日）

[2] 曽根直人 (2012) 情報基盤センターにおける時系列データ活用，鳴門教育大学情報教育ジャーナル，No.9，pp.37-42

[3] TimescaleDB, https://docs.timescale.com/timescaledb/latest/ （最終アクセス日：2021 年 1 月 7 日）

[4] influxDB, https://www.influxdata.com/products/influxdb/ （最終アクセス日：2021 年 1 月 7 日）