OpsBear
Towards the Ops Autonomy

Zabbixによる
収集データの効果的活用

運用自律化に向けたデータ分析について考える

TIS株式会社

池田 大輔

いけだ　　だいすけ

池田　大輔

TIS株式会社

IT基盤エンジニアリング第1部

@ike_dai　　ikedai

・2012年,2015年 Zabbix Conf@Riga登壇
・2013年,2015年,2016年 Zabbix Conf@Japan登壇
・2014年 Zabbix徹底活用本執筆

ike-dai　　ike_dai

OpsBear
Towards the Ops Autonomy..

収集した監視データをどう活かすか？

Zabbixにより
収集される監視データの内容って？

リソースの傾向情報

アプリケーションステータス情報

死活情報

Zabbix

Syslog情報

アクセスログ情報

アプリケーションステータス情報

APP/MW/DB等ログ情報

OpsBear
Towards the Ops Autonomy.

数

数値(整数) 型

数値(浮動小数) 型

テキスト

文字列 型

ログ 型

テキスト 型

OpsBear
Towards the Ops Autonomy.

ヒストリ

監視結果の生データ

※全データ型に対応

トレンド

1時間毎の統計データ

・最大値
・最小値
・平均値
・合計個数

※数値データのみ

Zabbix server: CPU user time

Graph

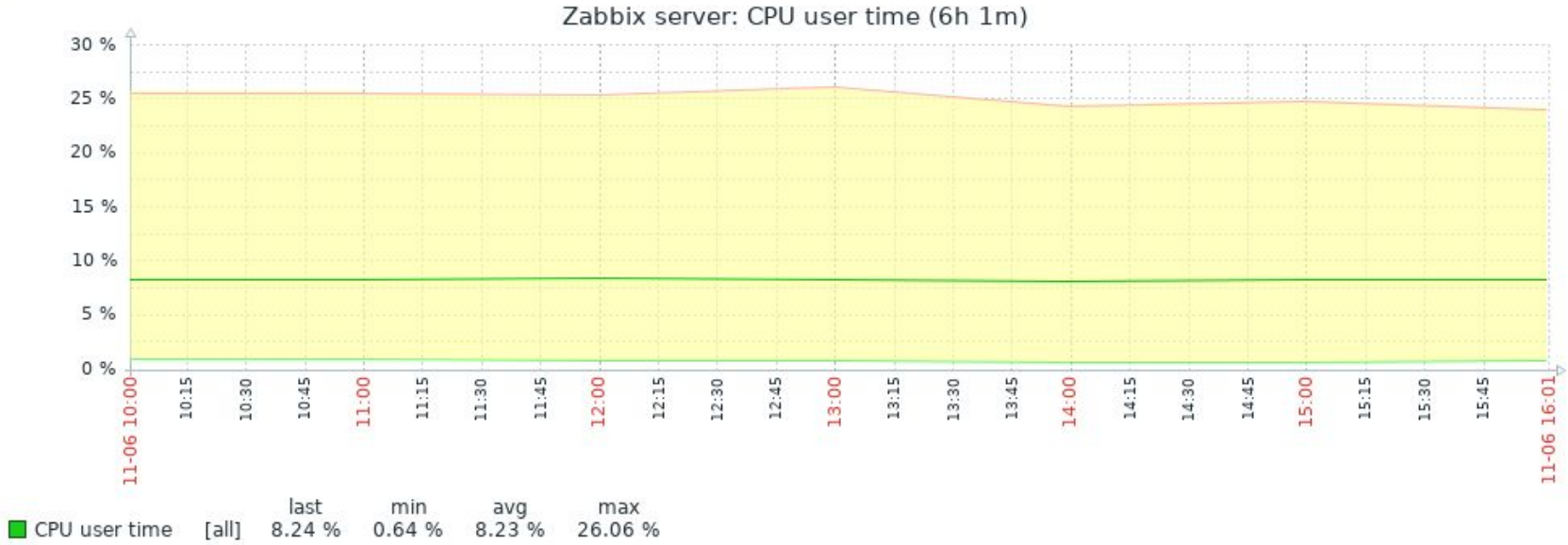
Filter ▲

Zoom: 5m 15m 30m 1h 2h 3h 6h 12h 1d 3d 7d 14d All

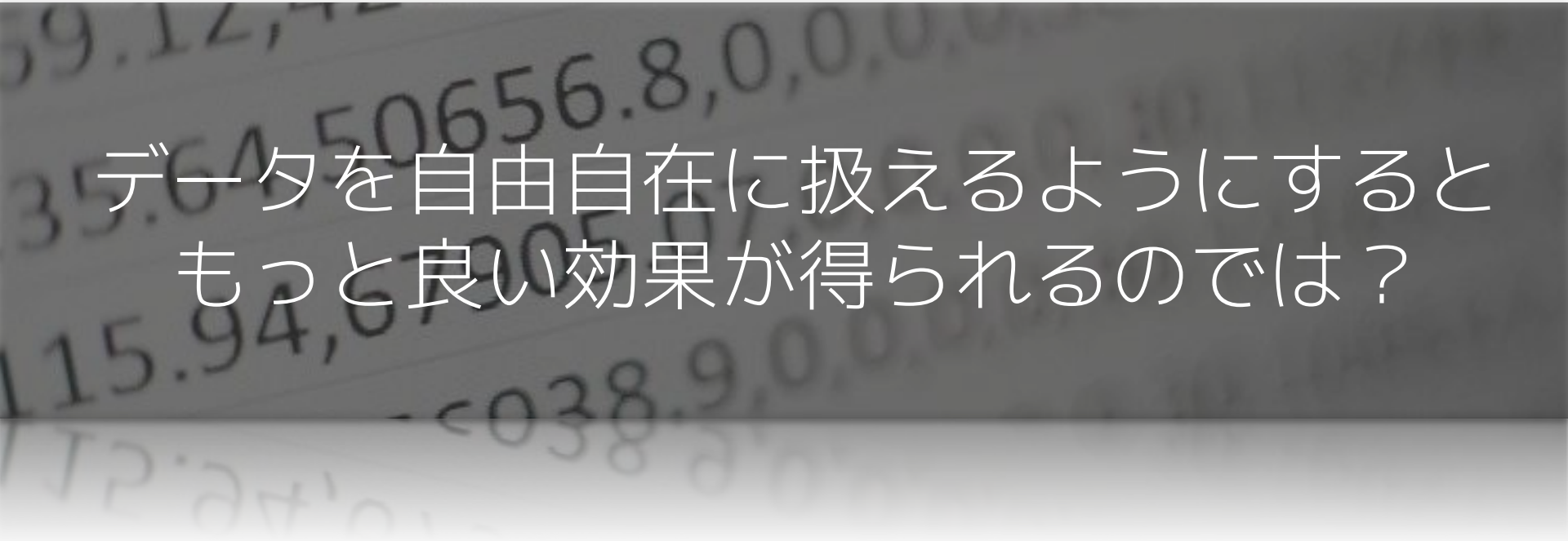
2018-11-06 10:00 - 2018-11-06 16:01

«« 7d 1d 12h 1h 5m | 5m 1h 12h 1d 7d »»

6h 1m dynamic



OpsBear
Towards the Ops Autonomy..



データを自由自在に扱えるようにすると
もっと良い効果が得られるのでは？

データ抽出方法の戦略

- Zabbix API機能
- Elasticsearchへの保存機能 (3.4.5以降)
- リアルタイムエクスポート機能 (4.0以降)
- Loadable Moduleによる書出機能 (3.2以降)

Zabbix API機能

- JSON RPCのWebAPI
- 監視設定の取得/監視設定の実施/監視結果の取得に対応
- Python, Ruby, Golang等各種ライブラリあり

ヒストリデータを取りたければ **history.get**

トレンドデータを取りたければ **trend.get**

Zabbix API機能

-
- 実施/監視結果
- ライブラリあ

history

trend.g

Request

```
{
    "jsonrpc": "2.0",
    "method": "history.get",
    "params": {
        "output": "extend",
        "history": 0,
        "itemids": "23296",
        "sortfield": "clock",
        "sortorder": "DESC",
        "limit": 10
    },
    "auth": "12fae.....af334",
    "id": 1
}
```

Response

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23296",
            "clock": "1351090996",
            "value": "0.0850",
            "ns": "563157632"
        },
        {
            "itemid": "23296",
            "clock": "1351090456",
            "value": "0.2750",
            "ns": "435307141"
        }
    ],
    "id": 1
}
```

Elasticsearchへの保存機能

- 特定のデータ型の監視データをElasticsearchに直接保存可能に
- Elasticsearchに保存したデータはZabbixのRDBMSには保存されない
- トレンドデータは扱われない

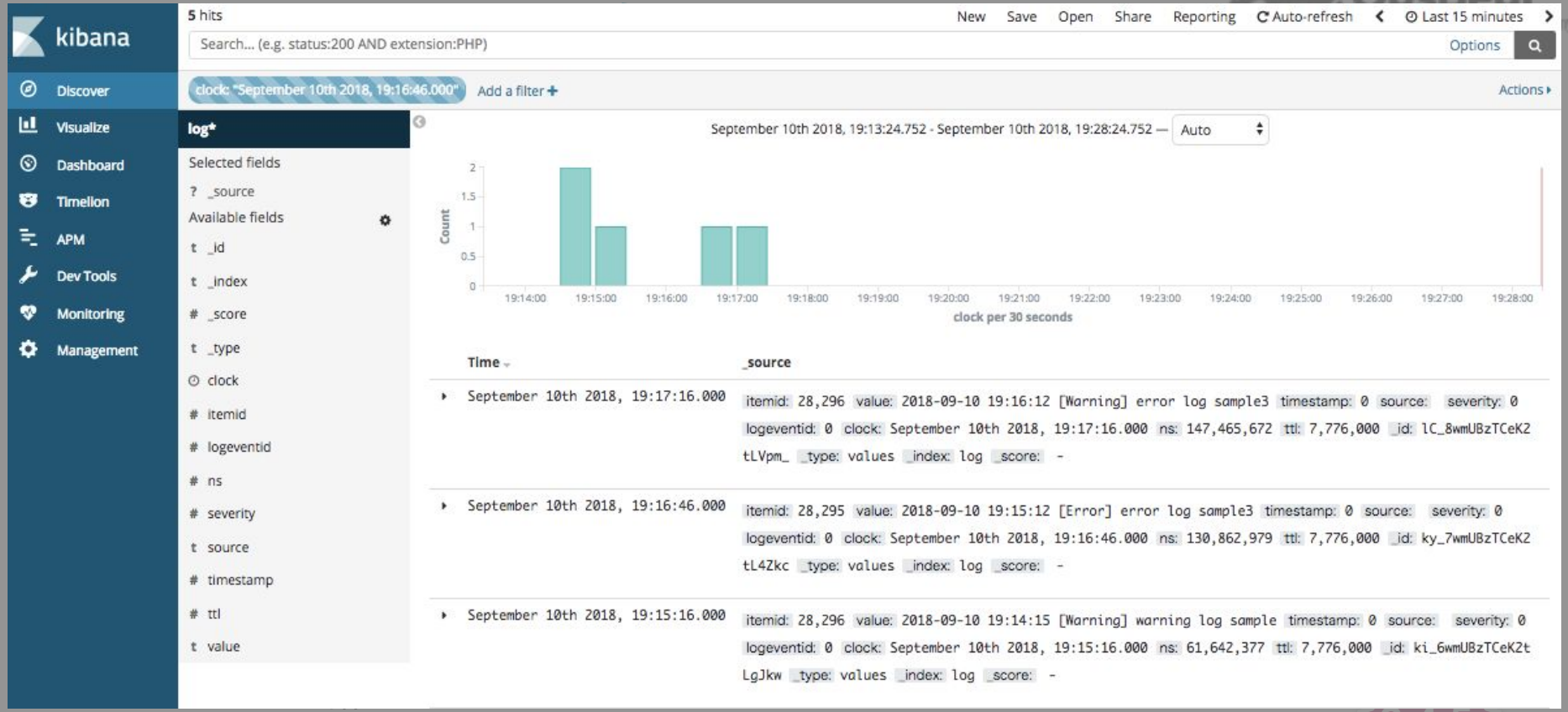
zabbix_server.conf

```
HistoryStorageURL=http://Elasticsearchのホスト名orIP:9200
HistoryStorageTypes=log,text
```

zabbix.conf.php

```
global $DB, $HISTORY; ←元のglobal $DB;の行を更新
・・・略
$HISTORY['url']   = 'http://Elasticsearchのホスト名orIP:9200';
$HISTORY['types'] = ['text', 'log'];
```

3.4.5 以降




```
$HISTORY['url']   = 'http://Elasticsearchのホスト名orIP:9200';
$HISTORY['types'] = ['text', 'log'];
```

3.4.5 以降

Elasticsearchへの保存時の注意ポイント

OpsBear
Towards the Ops Autonomy.

- 格納時のfieldはRDB保存時とテーブルカラムと同様の項目になる
  - itemid
  - clock
  - value 等

- ログデータ等の場合、ログ内容をパースしてfield分割し、分析とかしたくなる

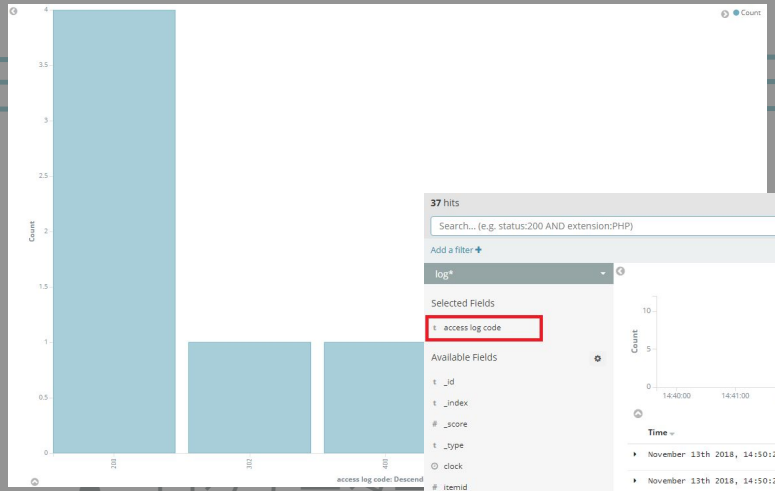
::1 - - [11/Nov/2018:03:09:27 +0900] "OPTIONS * HTTP/1.0" 200 - 72885 "-" "Apache/2.2.15 (CentOS) (internal dummy connection)"

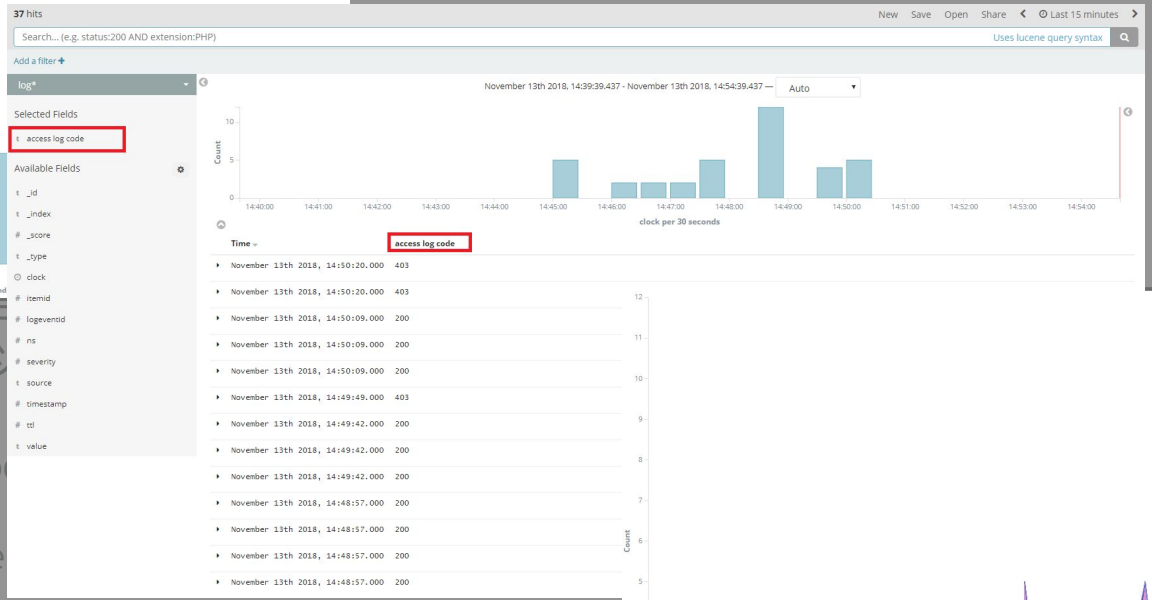
- その場合、以下のような処理を通すようにする必要がある
  - Ingest Nodeの**Grok processor**を使い**Pipeline前処理**でfield分割
  - **Script Field**を使って検索時に加工してfield追加

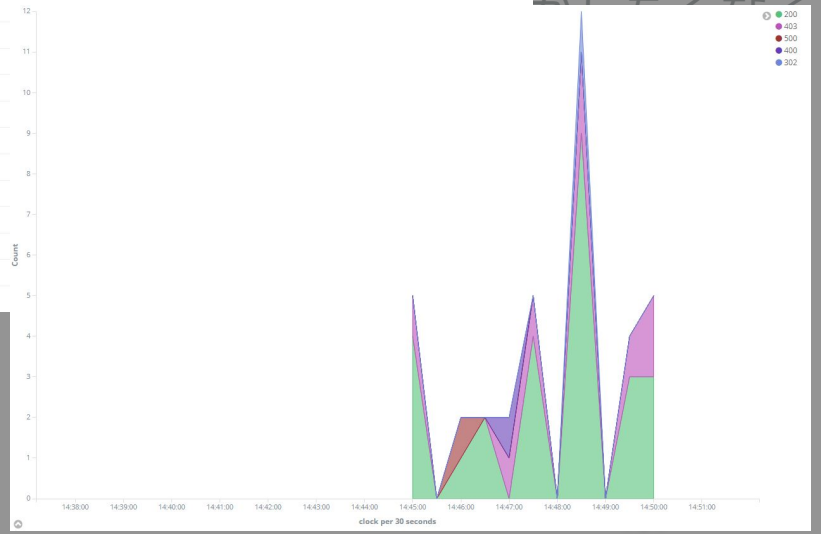
参考) https://qiita.com/ike_dai/items/b077ace4e87354afd8ca

3.4.5
以降

時の注意ポイント







- ログデータを
- その場合、以

  - Ingest No
  - Script Fie

リアルタイムエクスポート機能

- 監視データをテキストにリアルタイムに書き出し
- ヒストリ、トレンドだけでなくイベントデータも含め書き出し

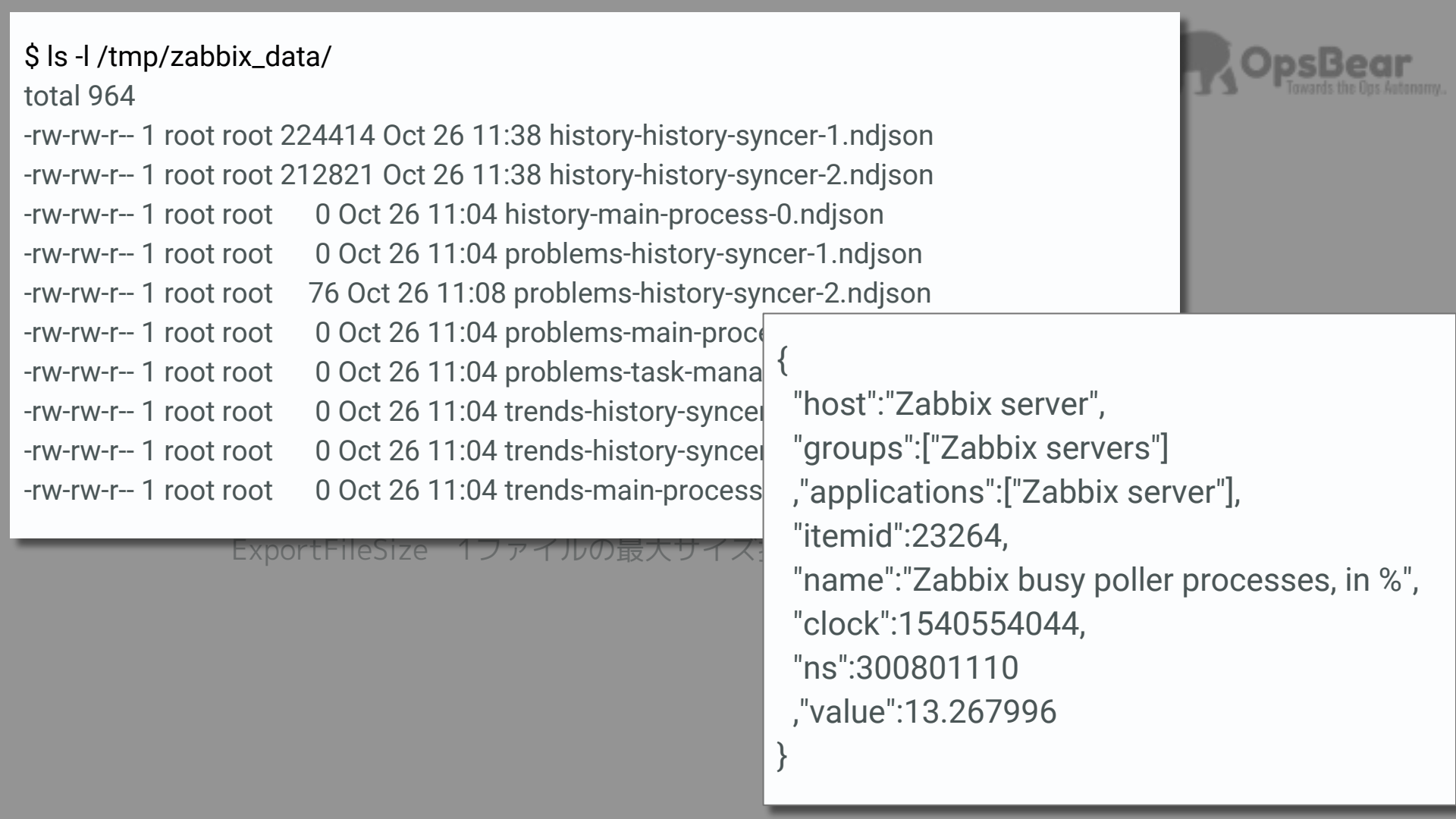
zabbix_server.conf
ExportDir　書き出し先のディレクトリを指定。
ExportFileSize　1ファイルの最大サイズ指定(1MB-1GB)。指定なしの場合1GB

4.0
以降

```
$ ls -l /tmp/zabbix_data/
total 964
-rw-rw-r-- 1 root root 224414 Oct 26 11:38 history-history-syncer-1.ndjson
-rw-rw-r-- 1 root root 212821 Oct 26 11:38 history-history-syncer-2.ndjson
-rw-rw-r-- 1 root root      0 Oct 26 11:04 history-main-process-0.ndjson
-rw-rw-r-- 1 root root      0 Oct 26 11:04 problems-history-syncer-1.ndjson
-rw-rw-r-- 1 root root     76 Oct 26 11:08 problems-history-syncer-2.ndjson
-rw-rw-r-- 1 root root      0 Oct 26 11:04 problems-main-process-0.ndjson
-rw-rw-r-- 1 root root      0 Oct 26 11:04 problems-task-manager-1.ndjson
-rw-rw-r-- 1 root root      0 Oct 26 11:04 trends-history-syncer-1.ndjson
-rw-rw-r-- 1 root root      0 Oct 26 11:04 trends-history-syncer-2.ndjson
-rw-rw-r-- 1 root root      0 Oct 26 11:04 trends-main-process-0.ndjson
```

ExportFileSize　1ファイルの最大サイズ指定(1MB-1GB)。指定なしの場合1GB

4.0
以降


$ ls -l /tmp/zabbix_data/
total 964
-rw-rw-r-- 1 root root 224414 Oct 26 11:38 history-history-syncer-1.ndjson
-rw-rw-r-- 1 root root 212821 Oct 26 11:38 history-history-syncer-2.ndjson
-rw-rw-r-- 1 root root 0 Oct 26 11:04 history-main-process-0.ndjson
-rw-rw-r-- 1 root root 0 Oct 26 11:04 problems-history-syncer-1.ndjson
-rw-rw-r-- 1 root root 76 Oct 26 11:08 problems-history-syncer-2.ndjson
-rw-rw-r-- 1 root root 0 Oct 26 11:04 problems-main-proce
-rw-rw-r-- 1 root root 0 Oct 26 11:04 problems-task-mana
-rw-rw-r-- 1 root root 0 Oct 26 11:04 trends-history-syncer
-rw-rw-r-- 1 root root 0 Oct 26 11:04 trends-history-syncer
-rw-rw-r-- 1 root root 0 Oct 26 11:04 trends-main-process
{
"host":"Zabbix server",
"groups":["Zabbix servers"]
,"applications":["Zabbix server"],
"itemid":23264,
"name":"Zabbix busy poller processes, in %",
"clock":1540554044,
"ns":300801110
,"value":13.267996
}

LoadableModuleによる書出機能

- Loadable Moduleとは、Zabbixの機能をCのプログラムで実装し、コンポーネント内に取り込むことができる機能(2.2から実装)
- 監視機能のカスタマイズ(新しい監視アイテムキーを実装する等)に有効
- ヒストリの書出し時処理をフックして自由に取り出すことができるように

3.2 以降

history_save_to_file.c

```
--
・・・略
static void         example_history_log_cb(const ZBX_HISTORY_LOG *history, int history_num)
{
        int         i;
        for (i = 0; i < history_num; i++)
        {
                FILE *file;
                file = fopen("./sample_log.txt", "a");
                fprintf(file, "itemid: %d, value: %s \n", history[i].itemid, history[i].value);
                fclose(file);
        }
}
・・・略
ZBX_HISTORY_WRITE_CBS       zbx_module_history_write_cbs(void)
{
        static ZBX_HISTORY_WRITE_CBS          example_callbacks =
        {
           ・・・略
                example_history_log_cb
        };
        return example_callbacks;
}
・・・略
```


し、

)に有効

きるように

3.2
以降

history_save_to_file.c

```
--
・・・略
static void		example_history_log_cb(const ZBX_HISTORY_LOG *history, int history_num)
{
	int	i;
	for (i = 0; i < history_num; i++)
	{
		FILE *file;
		file = fopen("./sample_log.txt", "a");
		fprintf(file, "itemid: %d, value: %s \n", history[i].itemid, history[i].value);
		fclose(file);
	}
}
・・・略
ZBX_HISTORY_WRITE_CBS	zbx_module_history_write_cbs(void)
{
	static ZBX_HISTORY_WRITE_CBS	example_callbacks =
	{
	   ・・・略
		example_history_log_cb
	};
	return example_callbacks;
}
・・・略
```

zabbix_server.conf

```
--
・・・略
LoadModulePath=/var/lib/zabbix/modules
LoadModule=history_save_to_file.so
・・・略
```

監視データをいろいろ応用する
イメージがわきましたか？

人依存になりがちな
しきい値ベース、キーワードベースの削減

OpsBear
Towards the Ops Autonomy.

しきい値ベース

CPU使用率が50%を超えたら

キーワードベース

logに「error」というキーワードが出たら

しきい値ベース

CPU使用率が**50%**を超えたら

どういう根拠で？
他は見なくていいの？

キーワードベース

logに「**error**」というキーワードが出たら

削減のための戦略

- 過去の結果との比較
- 統計分析による傾向値
- 時系列数値データ変化点・外れ値
- テキストログ出力量の時系列変化

等

削減のための戦略

Zabbix標準機能

- 過去の結果との比較
- 統計分析による傾向値

- 時系列数値データ変化点・外れ値
- テキストログ出力量の時系列変化

応用

等

過去の結果との比較

- Zabbixの**タイムシフト機能**の活用
- 過去のヒストリ結果をベースラインとした状態検知が可能
- タイムシフト機能はたいていのZabbixトリガー関数に実装

前日1日分の最大値を超える値を検知する例

{server-1:system.cpu.util[,system].last()}>{server-1:system.cpu.util[,system].max(**1d,1d**)}

前日1日分の90パーセンタイル値を超える値を検知する例

{server-1:system.cpu.util[,system].last()}>{server-1:system.cpu.util[,system].percentile(**1d,1d,90**)}

※トリガーで扱うデータが範囲が多くなるとValueCacheの枯渇には注意

統計分析による傾向値

- forecast関数、timeleft関数を活用
- ある統計モデルを仮定して推測した場合、指定した値に達するまでにどれぐらい時間がかかるか(timeleft)、n秒後にどのような値に達するか(forecast)を評価

ディスク空き率の1日後の値を予測する例(直近1週間のデータを元に推定)

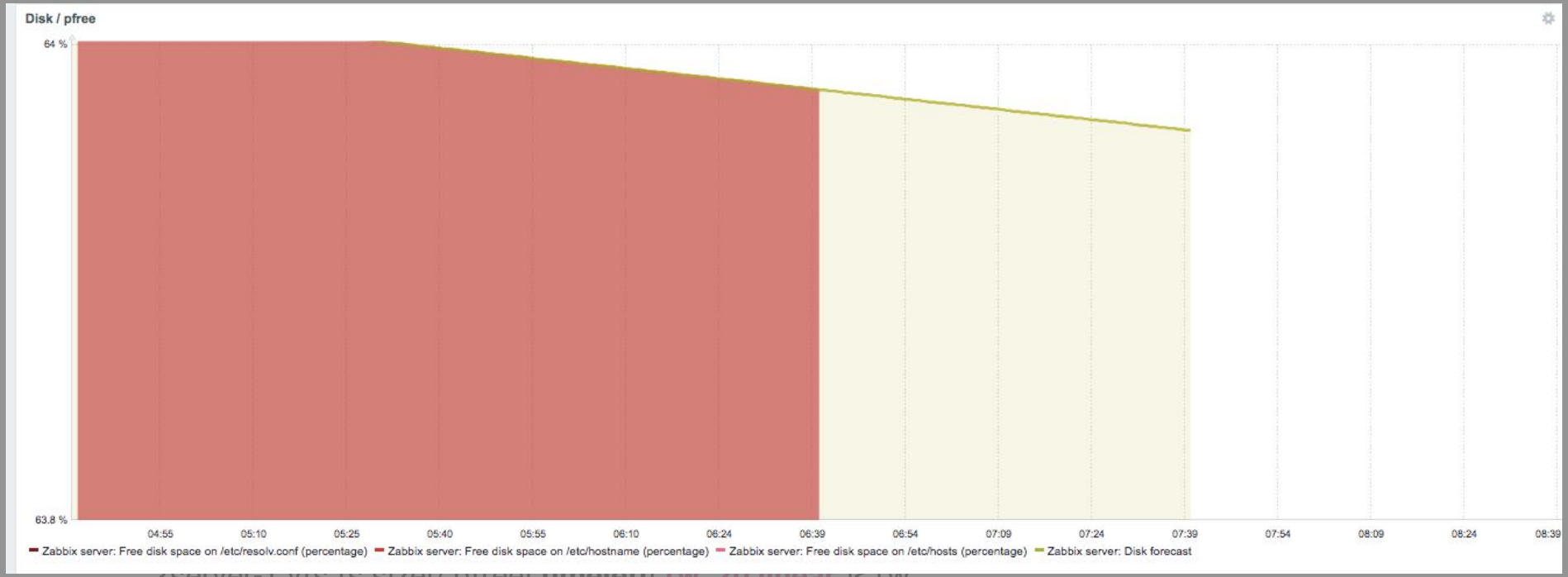
{server-1:vfs.fs.size[/,pfree].**forecast**(1w,,1d,linear,)<20

ディスク空き率が20%になるまでの残り時間を予測する例(直近1週間のデータを元に推定)

{server-1:vfs.fs.size[/,pfree].**timeleft**(1w,,20,linear,)<1w

統計分析による傾向値

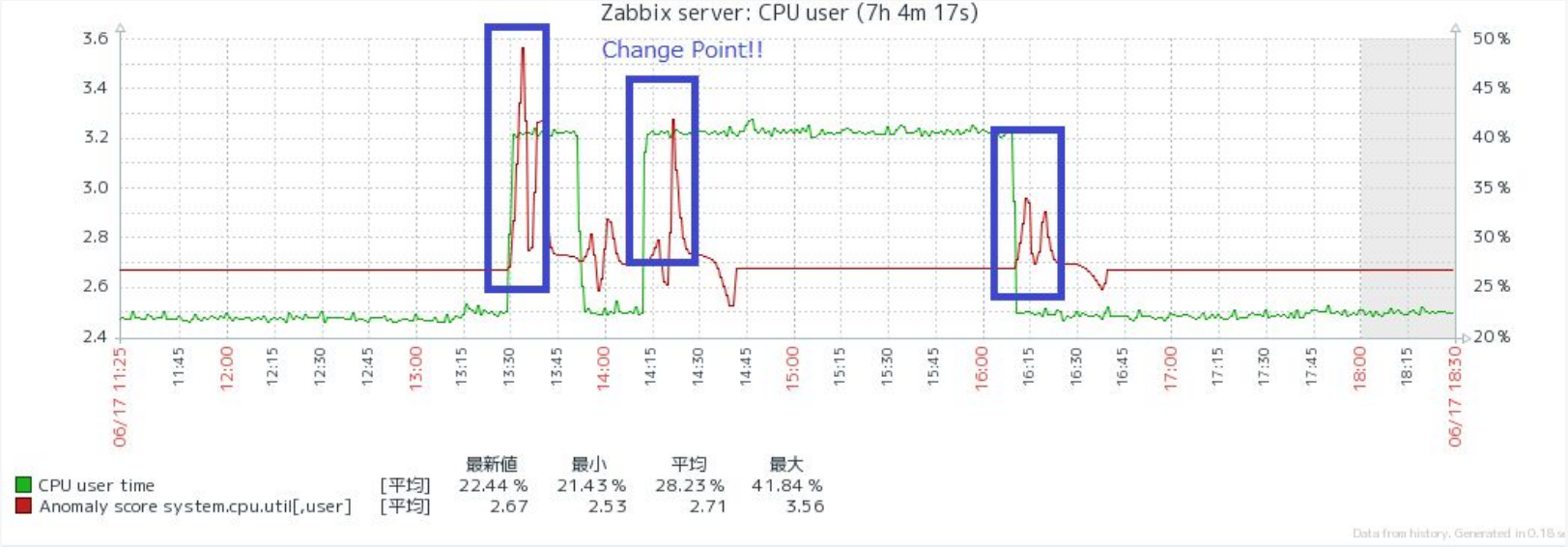
Zabbix4.0ダッシュボードのグラフ機能で未来時間に予測値をプロット表示可



{server-1:vfs.fs.size[/,pfree].timeleft(1w,,20,linear,)}<1w

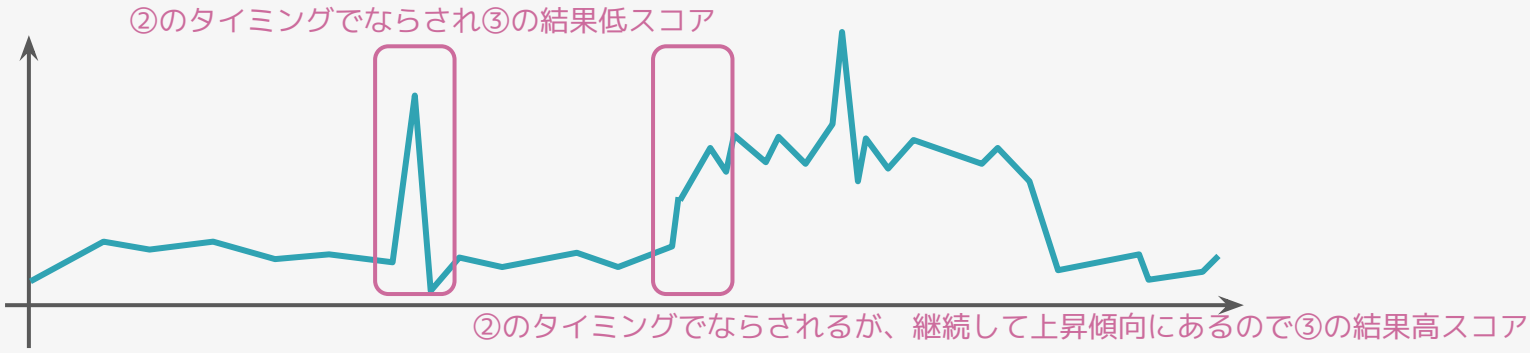
変化点・外れ値スコア

- 時系列で見て、突発的な異常(外れ値)
- 傾向の変わり目(変化点)を統計処理により検出



ChangeFinder

- SDARアルゴリズムをベースにした変化点検出アルゴリズム
  - ① 過去の傾向値を元に外れ値度合いを算出
  - ② 外れ値度合いの推移を平滑化
  - ③ 平滑化された外れ値度合いに対して更に外れ値度合いを算出



瞬間的に突出した値は変化とはみなさず
傾向が変わったタイミングを検出できる

ChangeFinder

Zabbix Anomaly Detector Plugin

GitHub: https://github.com/ike-dai/zabbix_anomaly

- Zabbix APIを通してhistory.getで値を取得
- 変化点スコアを算出してZabbixのヒストリとして登録
- スコアの高さを過去の傾向と比較して変化点異常を検出

傾向が変わったタイミングを検出できる

テキストログ出力量の時系列変化

- 単純な発生件数カウント(Zabbix標準のlog.countアイテムで可)
- 発生ログの内容を解析して分析

Elasticsearchの活用を考えてみる

強力なAggregation機能

Metric

検索結果の値の集計

Max, Min, Avg, Sum, Value count, Top hits,
Percentiles, Geo centroid, 等

Bucket

検索結果のキー毎の集計

Terms, SiginificantTerms, Range, IPv4Range,
GeoDistance, GeoHash, Filter, Data Histogram 等

Pipeline

Bucket内のデータのさらなる集計

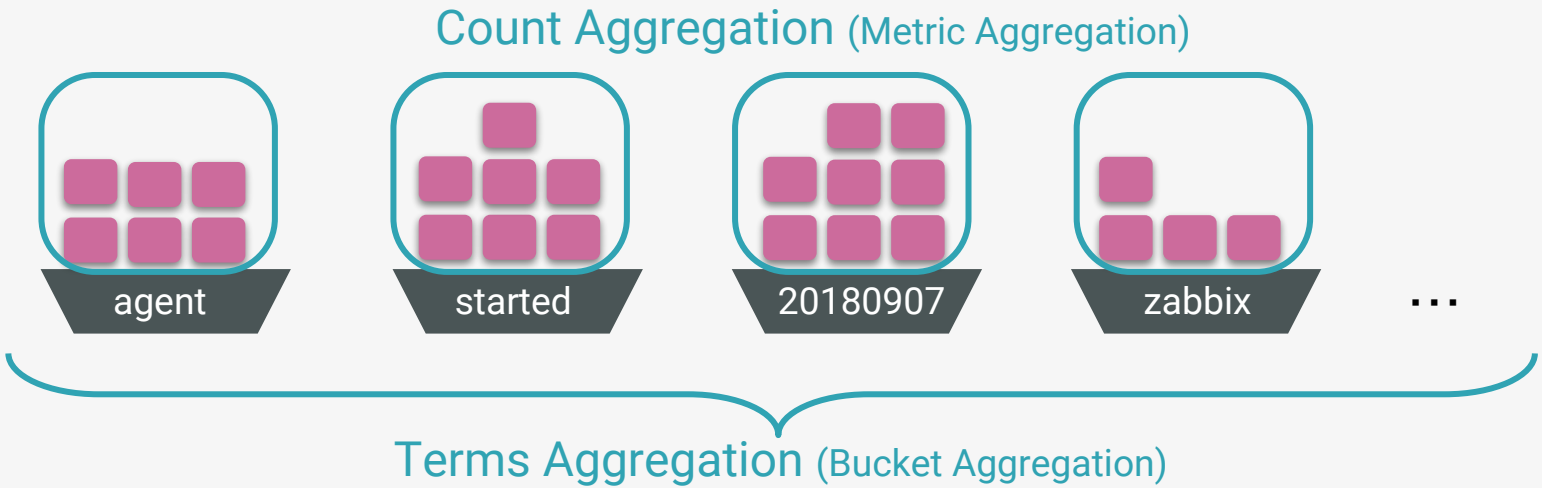
Avg Bucket, Max Bucket, Min Bucket, Sum Bucket,
Moving Avg Bucket, Derivative Bucket 等

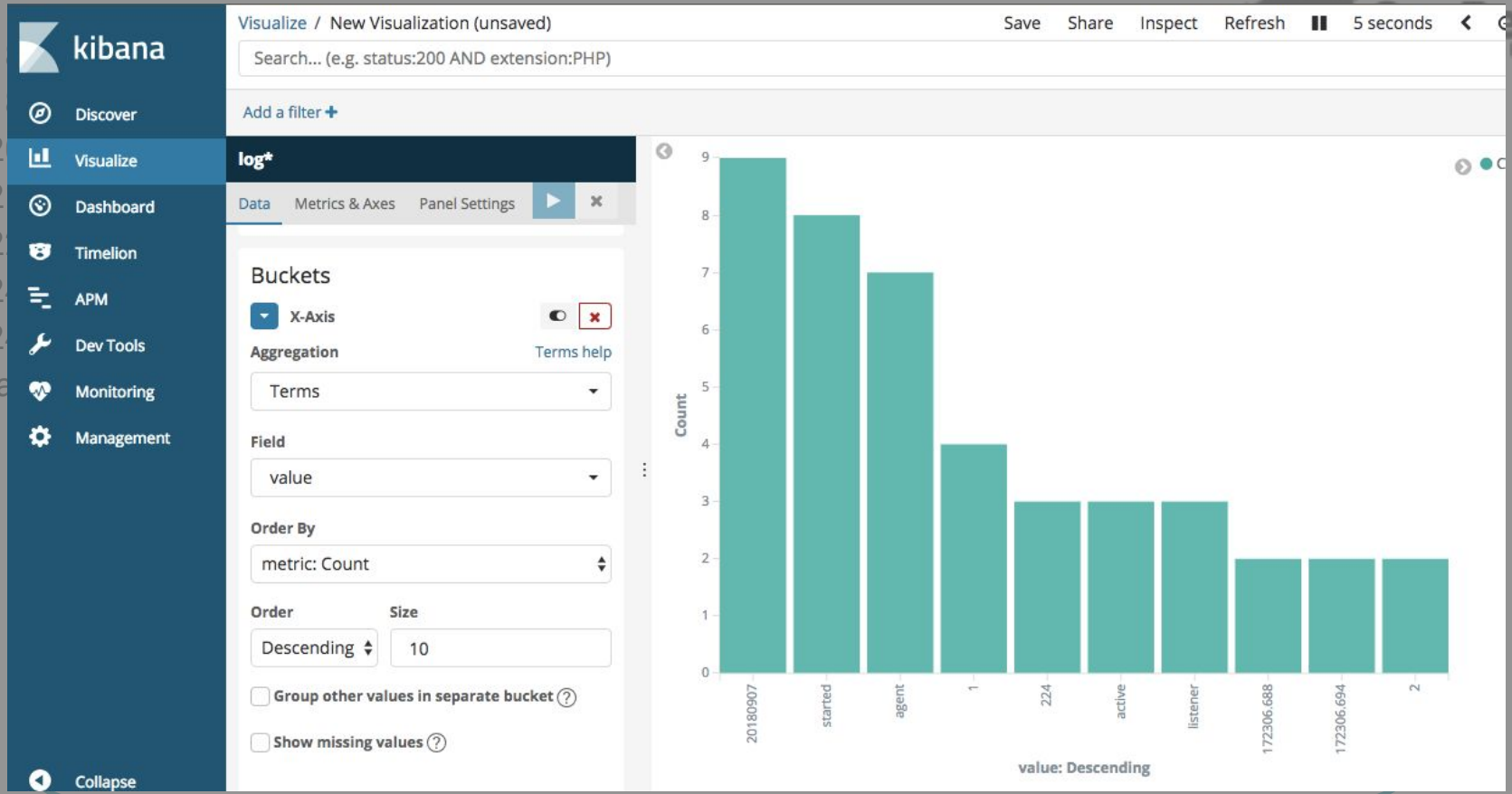
Matrix

複数のデータ間の関係分析

Matrix Stat
(データ間の相関・共分散、各データの統計)

218:20180907:172306.684 using configuration file: /etc/zabbix/zabbix_agentd.conf
218:20180907:172306.688 agent #0 started [main process]
220:20180907:172306.688 agent #1 started [collector]
221:20180907:172306.689 agent #2 started [listener #1]
222:20180907:172306.691 agent #3 started [listener #2]
224:20180907:172306.694 agent #5 started [active checks #1]
224:20180907:172306.696 active check configuration update from [127.0.0.1:10051] started to fail (cannot connect to [[127.0.0.1]:10051]: [111] Connection refused)





Terms Aggregation (Bucket Aggregation)

テキストデータも定量的に示せる**数**に
表現を変えることで見えてくることも

ログ出力パターンクラスタリング

218:20180907:172306.684 using configuration file: /etc/zabbix/zabbix_agentd.conf

n次元の特徴ベクトルに変換
[1,0,1,1,0,1,0,0,.....]

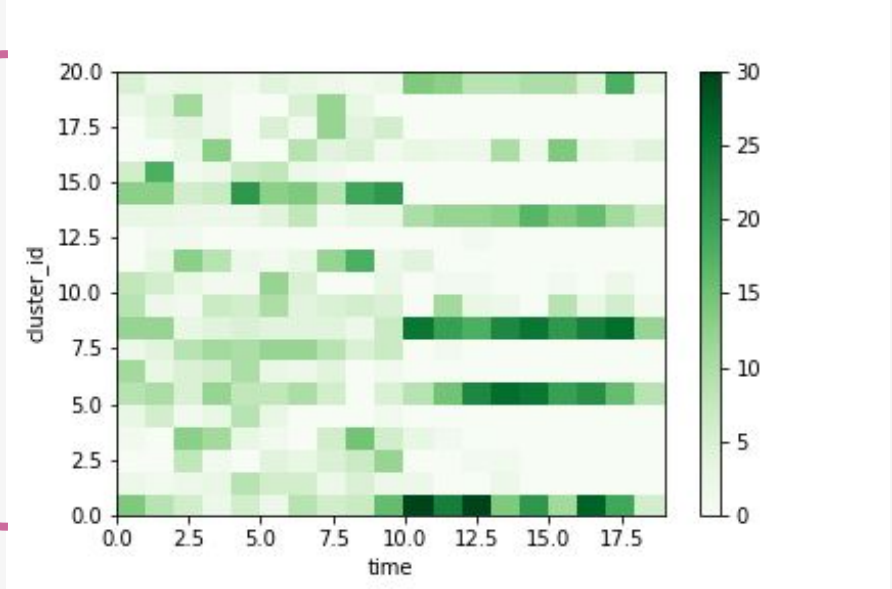
特徴ベクトルをクラスタリング

各クラスタごとの
発生頻度状況の変化を分析

異常の検知

クラスタ毎の時系列出力件数の推移に変換

ログ出力パターンで
クラスタリング



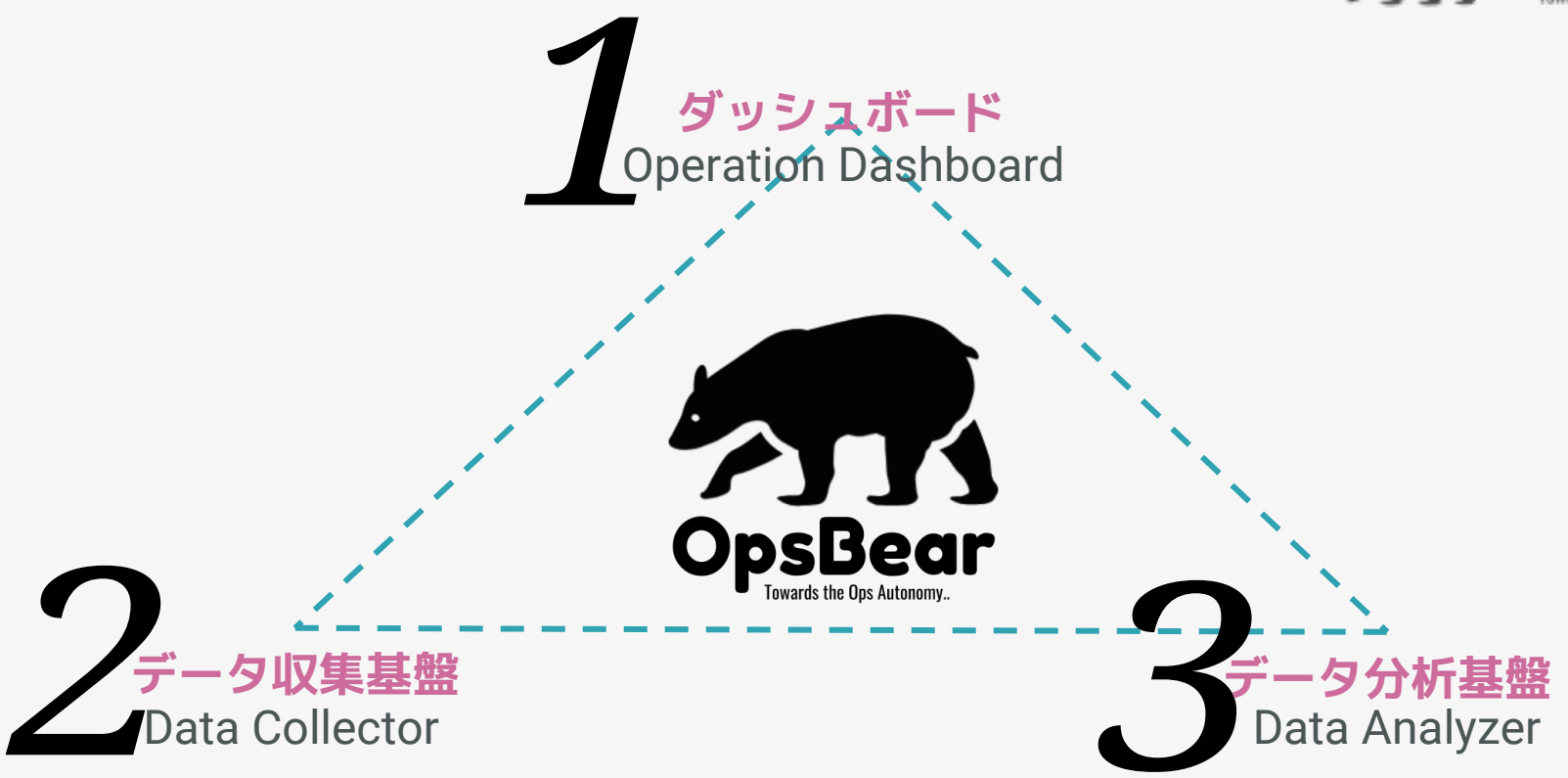

監視データを有効活用し、運用者の
調査・検討作業をサポートする仕組みを開発中

OpsBear
Towards the Ops Autonomy..

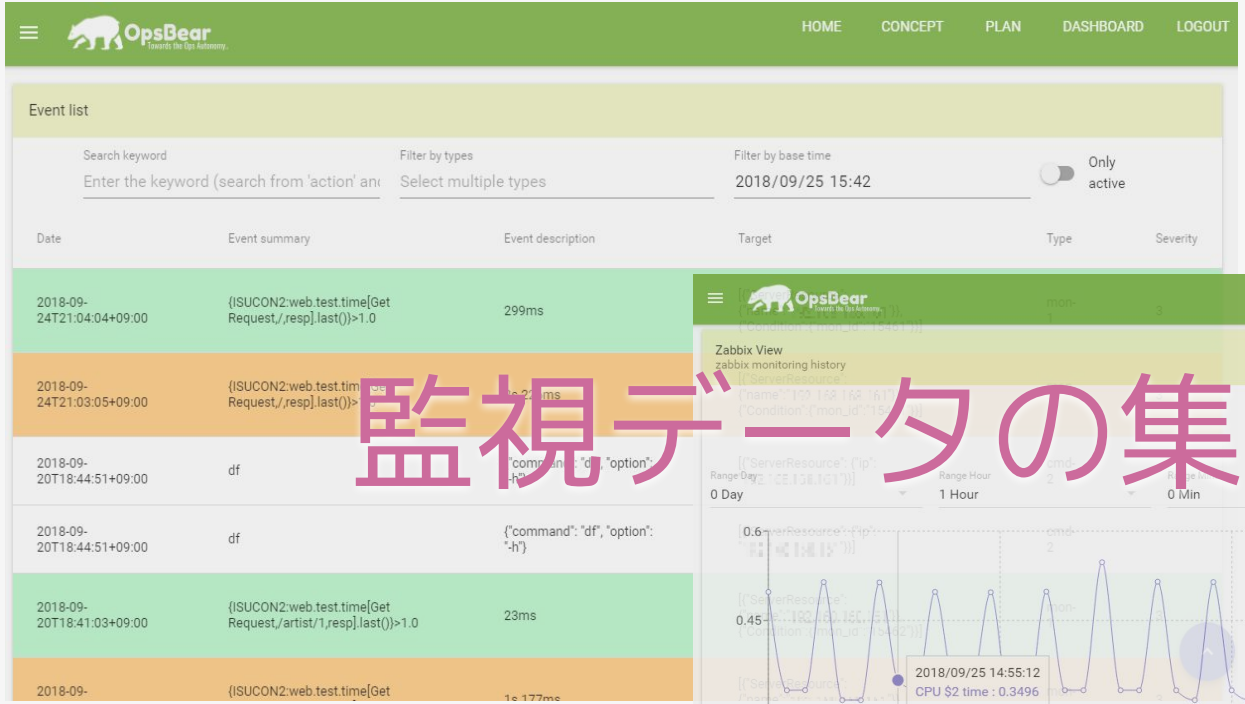
運 用

データ

運用現場にデータ分析のPowerを取り込む

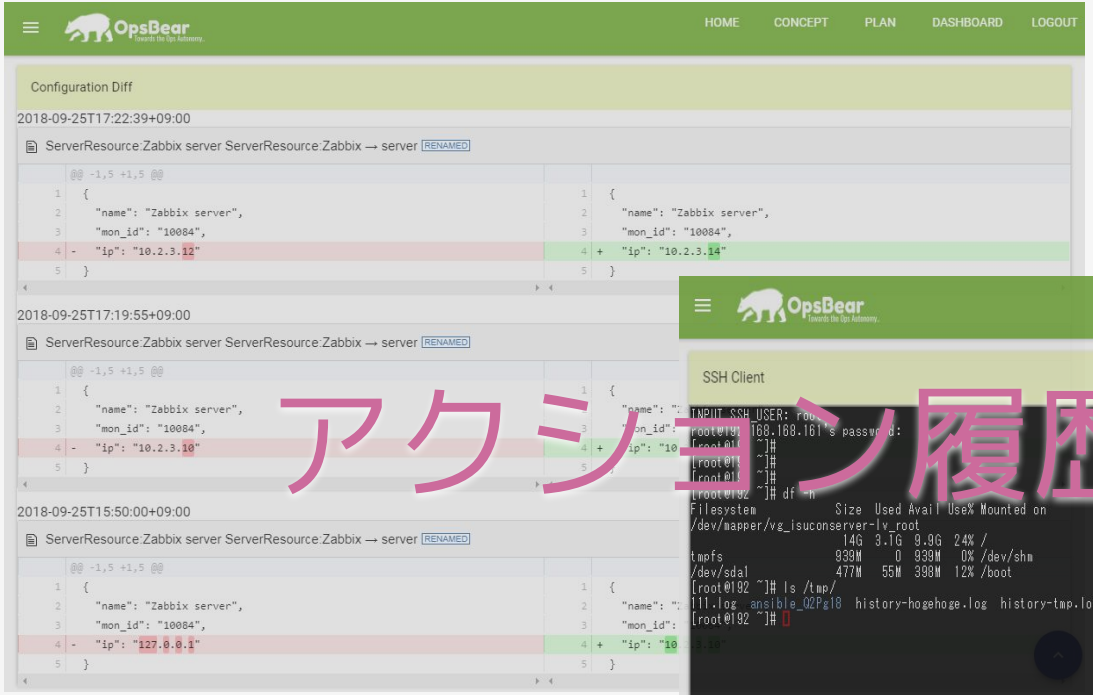

1
ダッシュボード
Operation Dashboard
OpsBear
Towards the Ops Autonomy..
2
データ収集基盤
Data Collector
3
データ分析基盤
Data Analyzer

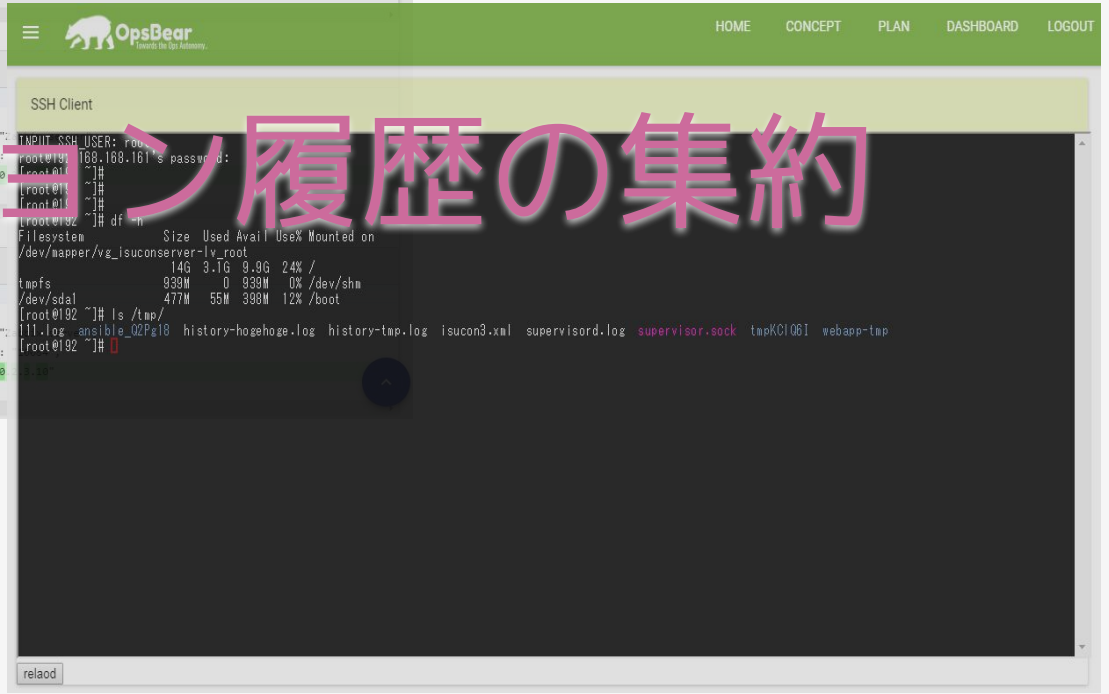
OpsBear
Towards the Ops Autonomy.



監視データの集約

OpsBear
Towards the Ops Autonomy





アクション履歴の集約

OpsBear
Towards the Ops Autonomy..

状態変化  行動実績

Next Action がわかる

現状に対し
何をすれば
改善する可能性があるか

OpsBear
Towards the Ops Autonomy..

興味あるかた一緒に取り組みませんか？


TIS
TIS INTEC Group