アノマリー検知とベースライン監視

ZABBIX 2021 Conference JAPAN

Zabbix Japan LLC
代表 寺島広大

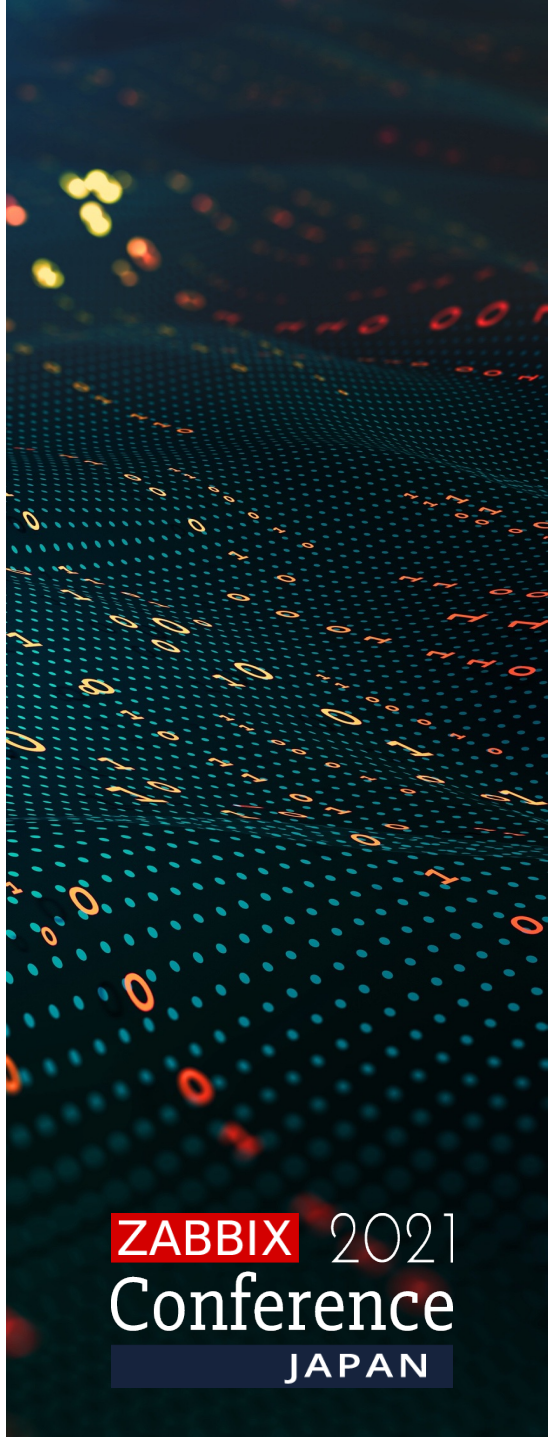
#ZabConfJp2021

6.0の新しい障害検知

- アノマリー検知とベースライン監視
  - ✓過去のデータから「通常と異なる」ことを検知する
  - ✓過去のデータの繰り返し期間の平均などを計算する

- 開発の状況
  - ✓アノマリー検知は6.0の次のalpha
  - ✓ベースライン監視はbetaまでに
  - ✓つまり、まだ利用できません

Zabbixの障害検知



トリガー設定の「条件式」

- 決まったフォーマットで障害条件を設定する
  - ✓この条件に一致したら障害検知

```
{ホスト名:アイテムのキー.トリガー関数}>70
```

- 復旧の条件も個別に指定できる
  - ✓この条件に一致したら復旧

```
{ホスト名:アイテムのキー.トリガー関数}<30
```

一般的な障害検知

- 閾値
  - ✓定数を決めて上回った場合/下回った場合
  - ✓CPU、メモリ、ディスクなどのリソース監視に利用する

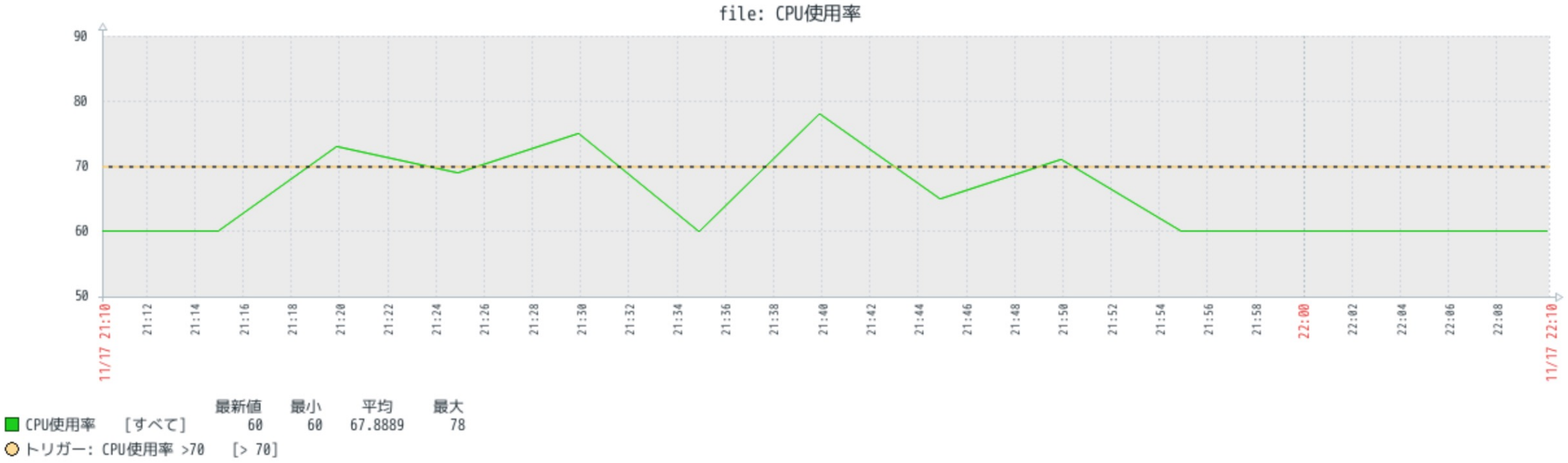
```
{hostname:system.cpu.util[,user].last()}>70
```

- 文字列一致
  - ✓監視値に指定した文字列が含まれていた場合
  - ✓ログ監視やSNMPトラップ監視に利用する

```
{hostname:log[/var/log/messages].str(CRIT)}=1
```

閾値の問題

- 70前後のデータが連続した場合



トリガー関数の活用

- 一定時間の平均値で障害検知

```
{hostname:system.cpu.util[,user].avg(1h)}>70
```

- 一定時間の閾値を超えた回数で障害検知

```
{hostname:system.cpu.util[,user].count(70,1h)}>5
```

- 複数の異なる監視設定を利用した障害検知

```
{hostname:system.cpu.util[,user].last()}>70 and
{hostname:system.cpu.load.last()}>5
```

タイムシフト

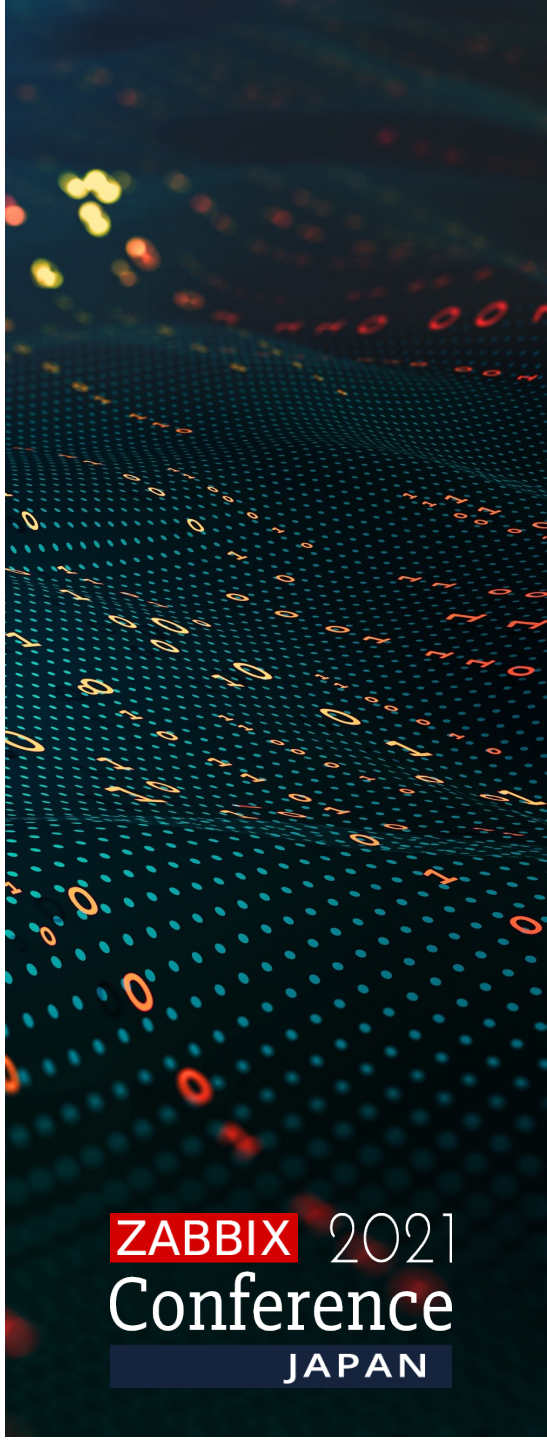
- 過去のデータと比較して障害検知
  - ✓1週間前の同じ時間帯よりトラフィックが2倍以上あれば障害

```
{hostname:net.if.in[eth0].avg(1h)} /
{hostname:net.if.in[eth0].avg(1h,1w)}>2
```

- 課題
  - ✓1週間前が祝日だった場合
  - ✓判定元の過去データが1つしか指定できない

障害の予測検知

Zabbix 3.0



forecastとtimeleftトリガー関数

- 過去のデータを元に、将来を予測する
  - ✓forecast: 指定した将来の時点の予測値を返す

```
forecast(sec|#num, <timeshift>, time, <fit>, <mode>)
```

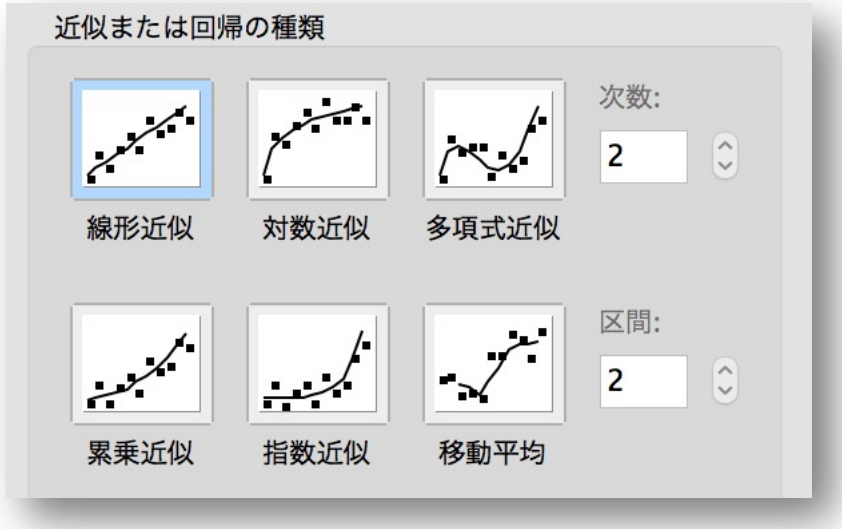
  - ✓timeleft: 指定した値になるまでの期間を返す

```
forecast(sec|#num, <timeshift>, threshold, <fit>)
```

forecast, timeshiftの統計関数

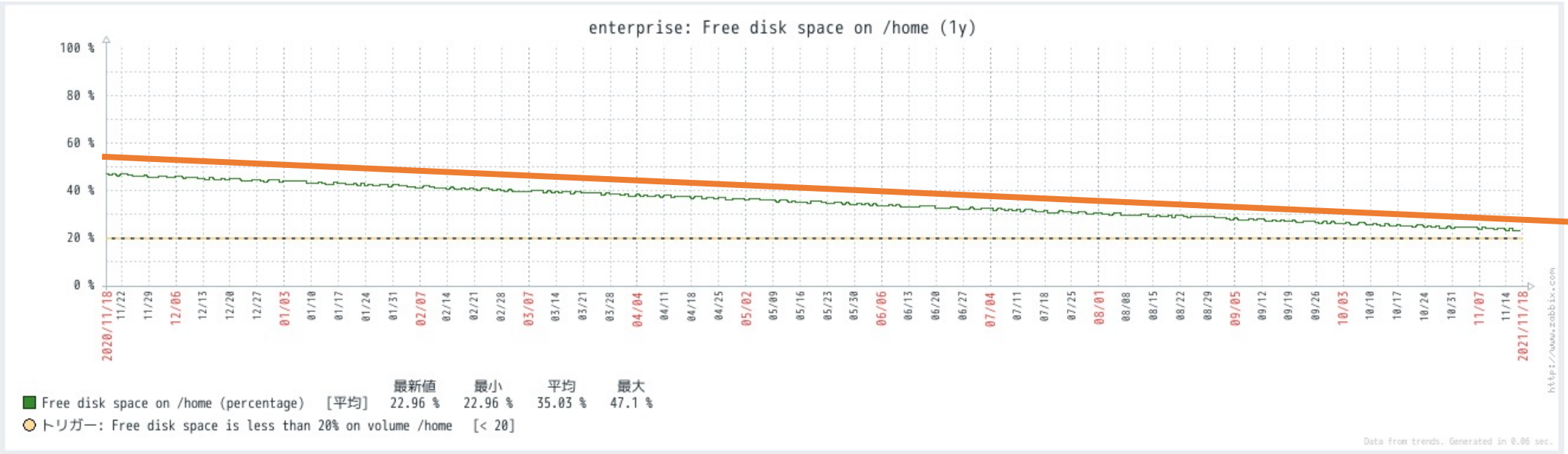
- fitパラメータに利用できる設定
  - ✓一般的な統計関数

パラメータ	統計関数
linear	線形近似
polynomialN	多項式近似
exponential	指数近似
logarithmic	対数近似
power	累乗近似



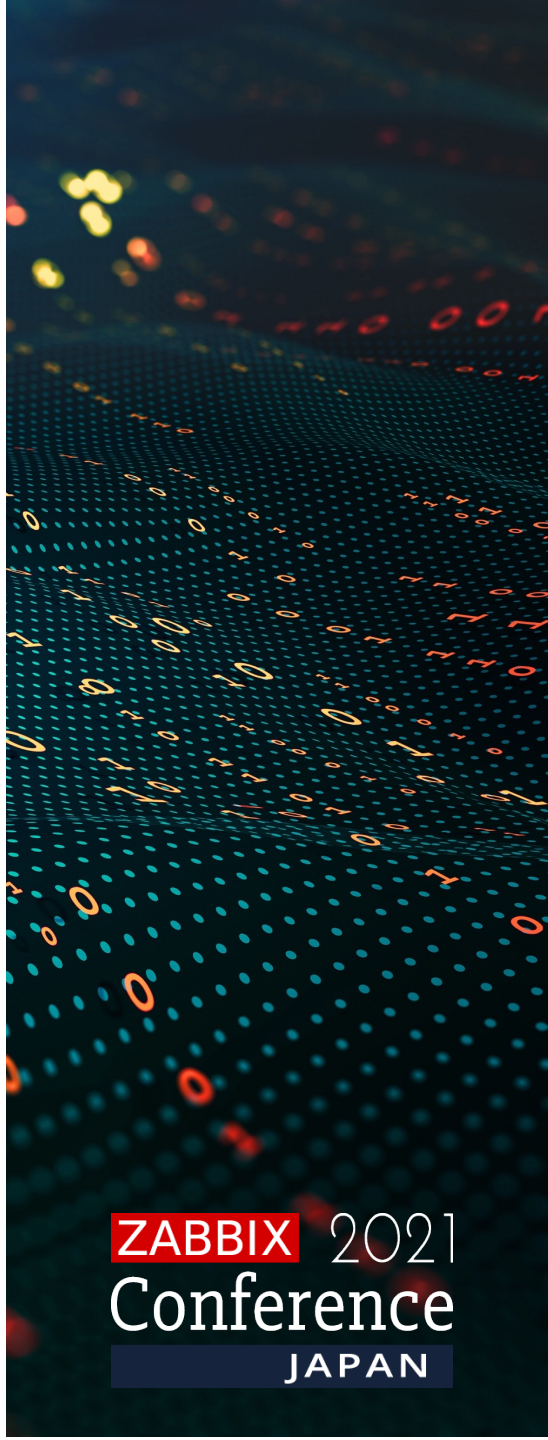
統計関数の使い方

- 基本はlinearを利用する



トレンド分析

Zabbix 5.2 & 5.4



トレンド分析

- 長期間の収集済み監視データを利用
  - ✓ヒストリデータではなくトレンドデータを利用
  - ✓長期間の過去データから平均/最大/最小/合計/個数を算出

- 1ヶ月の平均値が1ヶ月前より10%大きい場合に障害

```
{hostname:system.cpu.load.trendavg(1M,now/M)}>
1.1*{hostname:system.cpu.load.trendavg(1M,now/M-1M)}
```

トリガー条件式の記述の変更

- 5.2までのトリガー条件式

```
{hostname:system.cpu.util[,user].last()}>70
```

```
{hostname:log[/var/log/messages].str(CRIT)}=1
```

- 5.4以降のトリガー条件式

```
last(/hostname/system.cpu.util)>70
```

```
find(/hostname/log[/var/log/messages],"like","CRIT")=1
```

5.4以降のトリガー関数

6 Supported functions

Click on the respective function group to see more details.

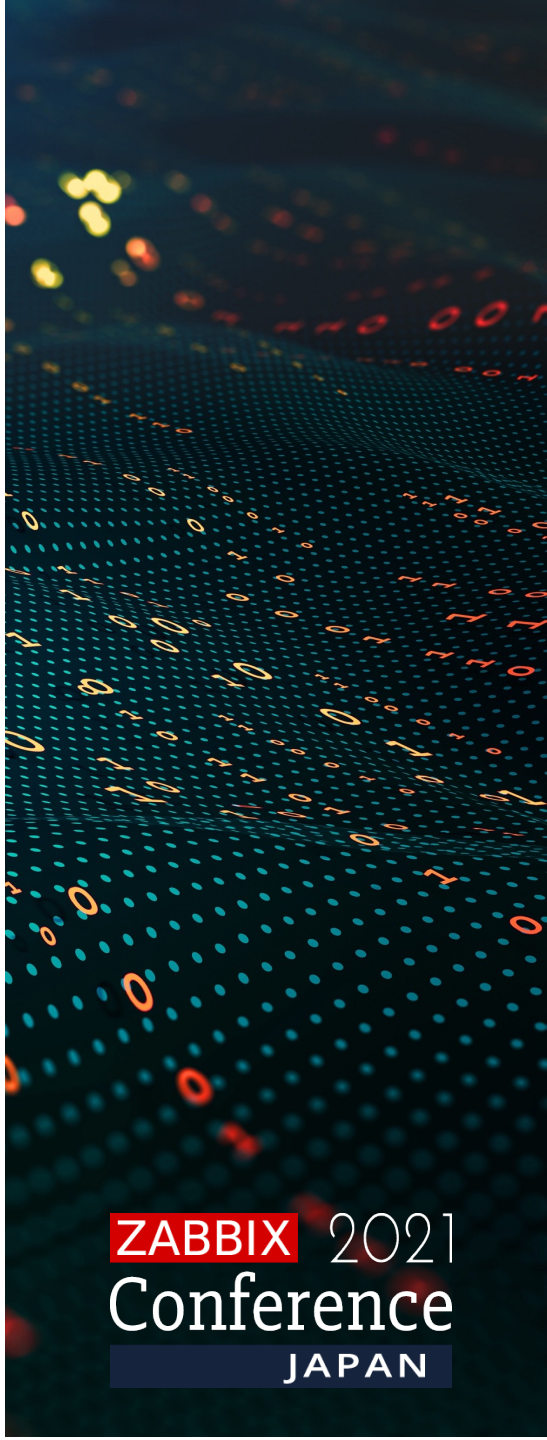
Function group	Functions
Aggregate functions	avg, kurtosis, mad, max, min, skewness, stddevpop, stddevsamp, sum, sumofsquares, varpop, varsamp
Bitwise functions	bitand, bitlshift, bitnot, bitor, bitrshift, bitxor
Date and time functions	date, dayofmonth, dayofweek, now, time
History functions	change, count, countunique, find, first, fuzzytime, last, logeventid, logseverity, logsource, nodata, percentile, trendavg, trendcount, trendmax, trendmin, trendsum
Mathematical functions	abs, acos, asin, atan, atan2, avg, cbrt, ceil, cos, cosh, cot, degrees, e, exp, expm1, floor, log, log10, max, min, mod, pi, power, radians, rand, round, signum, sin, sinh, sqrt, sum, tan, truncate
Operator functions	between, in
Prediction functions	forecast, timeleft
String functions	ascii, bitlength, bytelength, char, concat, insert, left, length, ltrim, mid, repeat, replace, right, rtrim, trim

These functions are supported in trigger expressions and calculated items.

https://www.zabbix.com/documentation/current/manual/appendix/functions/history

アノマリー検知と
ベースライン監視

Zabbix 6.0



MACHINE LEARNING: ZABBIX APPROACH

Easy and transparent:

- Simple configuration
- Easy to understand
- Easy to verify

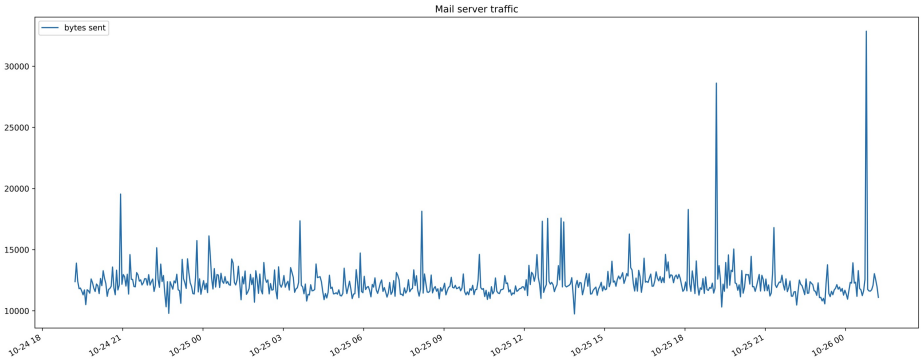
これまでの障害検知と異なること

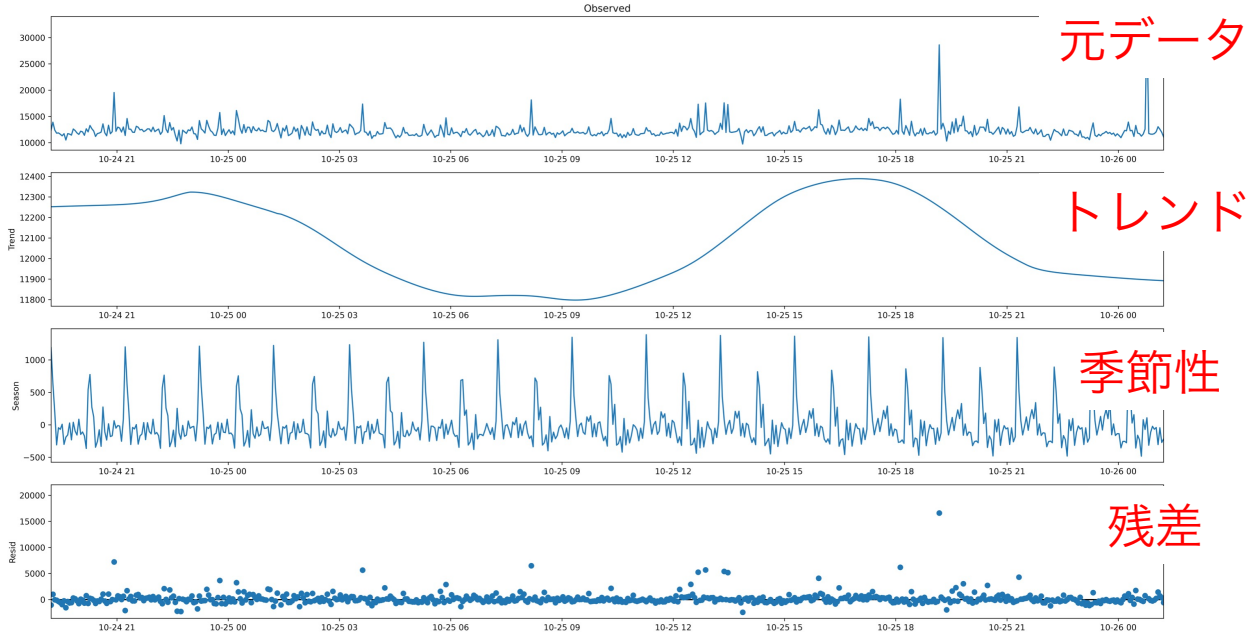
- これまで
  - ✓監視値(CPU使用率など)がどのような値になるかを数値で決める
  - ✓過去の値と比較して決める

- アノマリー検知とベースライン監視
  - ✓過去のデータと「異なる」＝「普段と違う」ことを検知
  - ✓閾値ではなく、過去のデータから計算した(学習した)値との違いの大きさを元に障害検知する

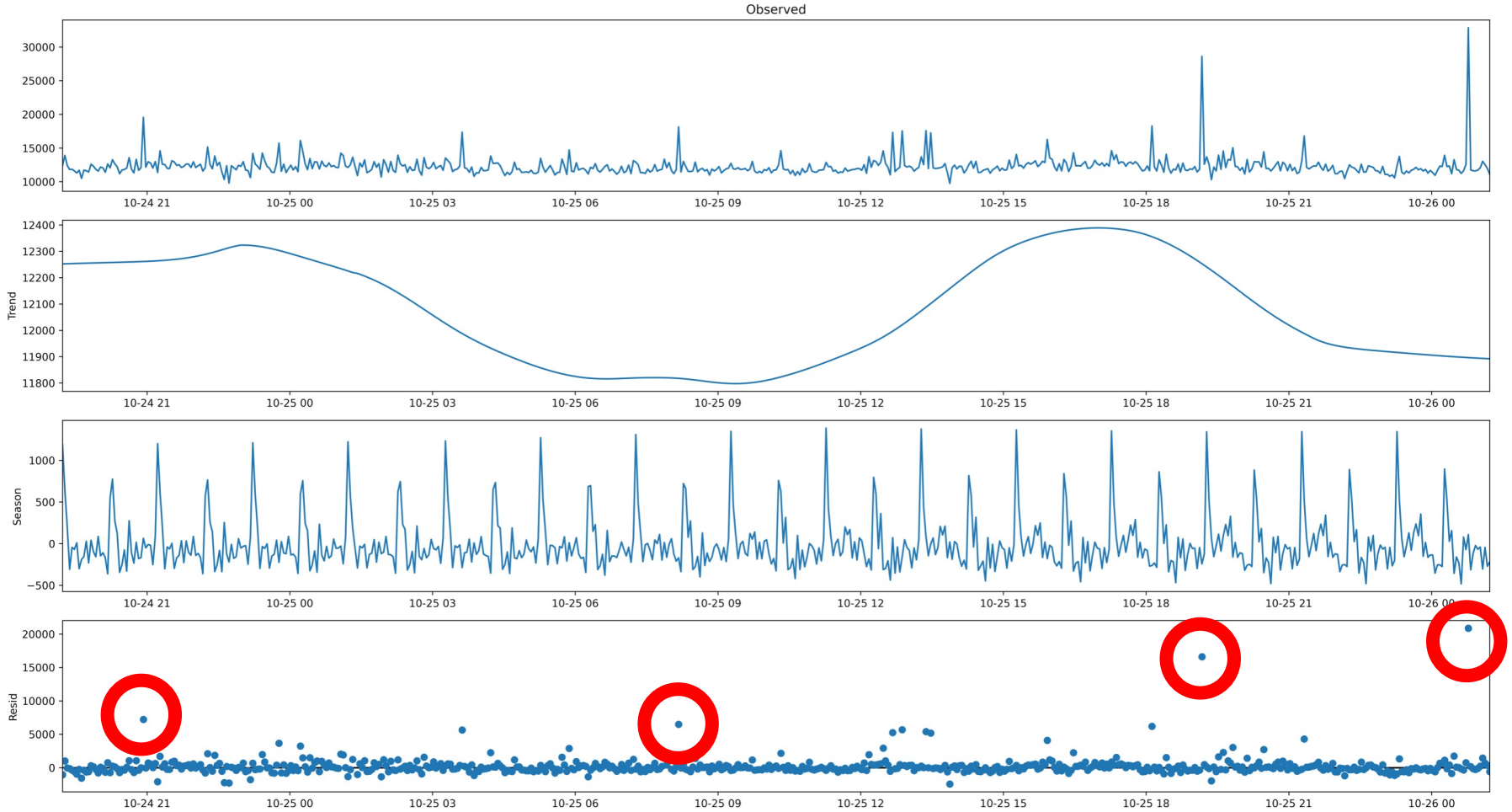
アノマリー検知

- 規則性や季節性のある監視データに利用できる
- トレンドデータから計算
- STL分解を利用





残差の偏差



元データ

トレンド

季節性

残差

アノマリー検知の関数

- trendstl関数
  - ✓全体のデータのうち異常値の数のレシオを0～1の間の値で返す

```
trendstl(/hostname/key,period:timeshift, detection period
                                ,season, deviations, dev algorithm)
```

パラメータ	説明
/hostname/key	ホスト名、アイテム
period:time	STL分解に利用するデータの期間
detection period	アノマリーを検知する期間
season	季節性の周期の期間
dev algorithm	STL分解のアルゴリズム

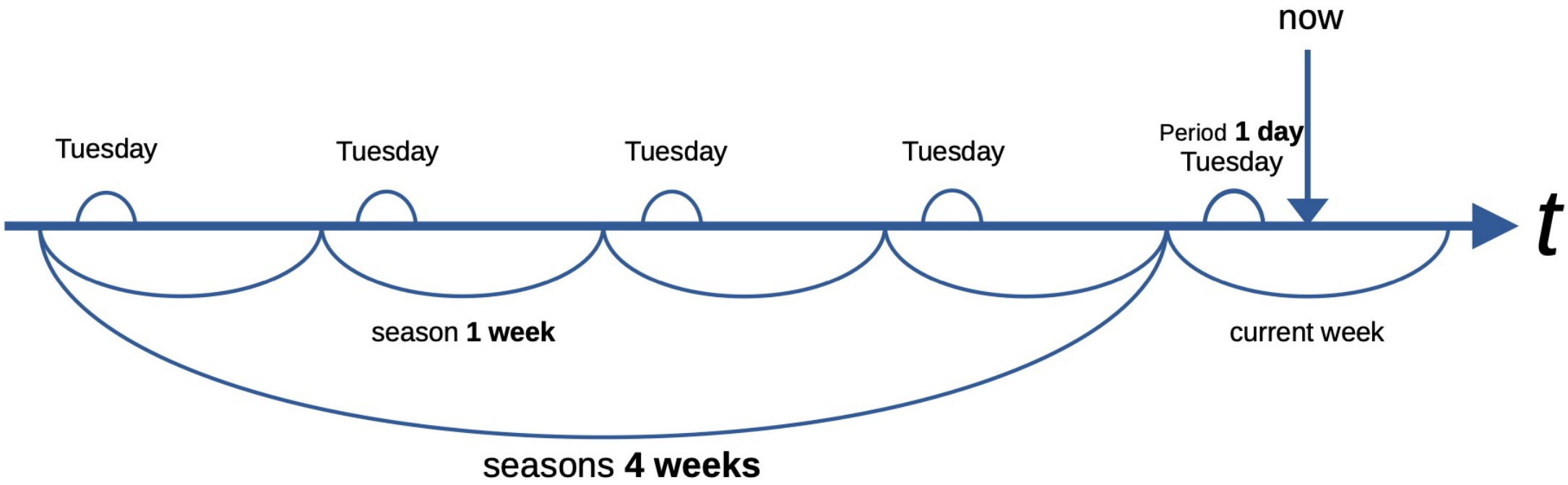
ベースライン監視

- 過去のデータから指定期間の繰り返しの平均値から得られる値
  - ✓トレンドデータから計算

- 「期間」と「周期」
  - ✓過去のカレンダーの複数期間の平均値
  - ✓過去4週間の毎月曜日
    - 期間：月曜日
    - 周期：4週間

期間と周期

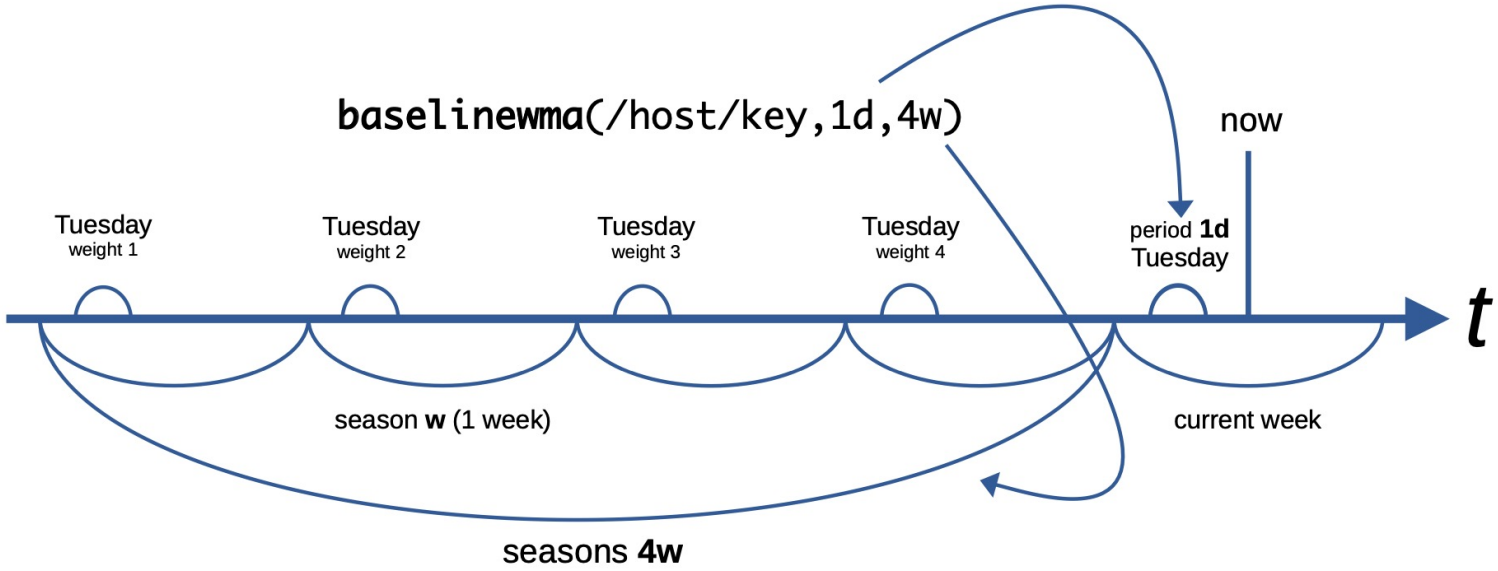
- 「期間: 1日」 「周期: 4週間」
  ✓現在が水曜日の場合、期間は前日の火曜日が利用される



ベースラインの関数

- baselinewma関数
  - ✓繰り返しの周期の中の指定した期間の平均値を返す

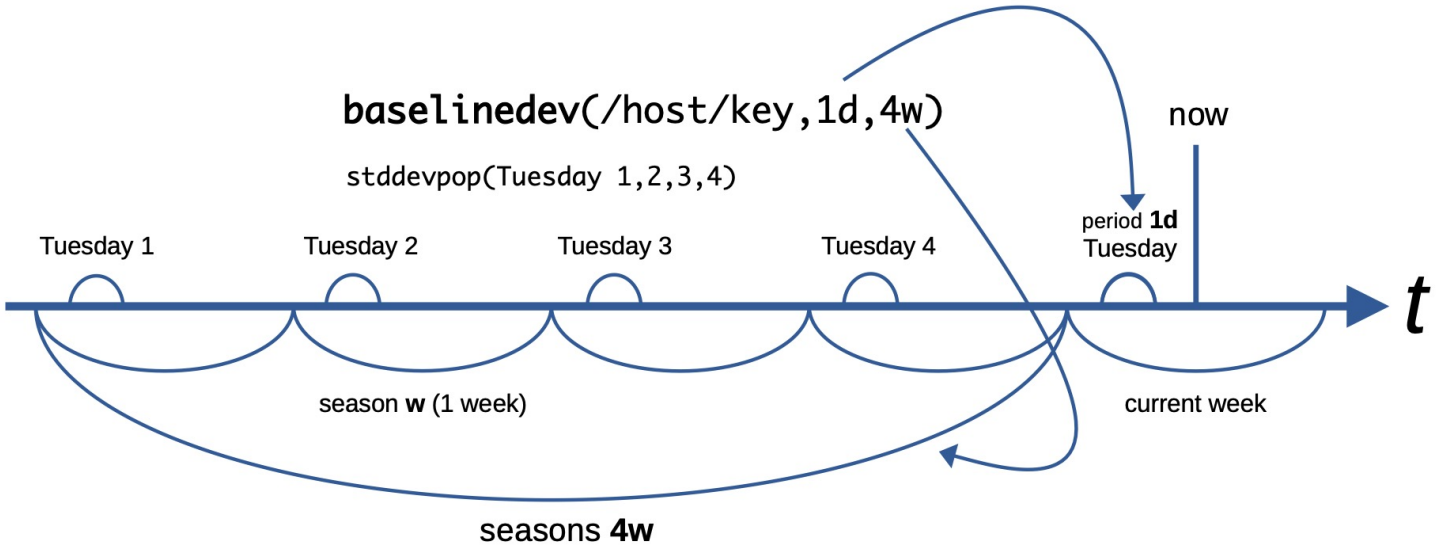
```
baselinewma(/host/key,period<:time shift>,seasons)
```



ベースラインの関数

- baselinedev関数
  - ✓繰り返しの周期の中の指定した期間の標準偏差を返す

```
baselinedev(/host/key,period<:time shift>,seasons)
```




Thank you
ZABBIX 2021
Conference
JAPAN