

## 8 Прогнозирующие функции триггеров

### Обзор

Иногда имеются признаки надвигающейся проблемы. Эти признаки можно заметить, поэтому можно заблаговременно предпринять меры, чтобы предотвратить или хотя бы уменьшить влияние проблемы.

В Zabbix имеются средства прогнозирования будущего поведения наблюдаемой системы на основе исторических данных. Эти средства реализованы в виде прогнозирующих функций триггеров.

### 13.9.1 Функции

Необходимо знать две вещи, а именно то, как описать состояние проблемы и сколько времени нужно, чтобы предпринять меры. Далее есть два способа создать триггер, сигнализирующий о возможной нежелательной ситуации. Первый: триггер должен “загореться”, когда система после “пора действовать”, как ожидается, будет в состоянии проблема. Второй: триггер должен “загореться”, если система перейдет в состояние проблемы за время меньше, чем “пора действовать”. Необходимо воспользоваться функциями триггеров, а именно **forecast** и **timeleft** соответственно. Надо заметить, что лежащий в основе статистический анализ, в основном идентичен у этих двух функций. Вы можете настроить триггер в зависимости от того, какой путь более предпочтителен для вас, и результаты будут идентичны.

### 13.9.2 Параметры

Обе функции используют почти одинаковый набор параметров. Обратитесь к списку [поддерживаемых функций](#) за справкой.

#### 13.9.2.1 Интервал времени

Прежде всего вам необходимо указать период истории, который Zabbix должен проанализировать для составления прогноза. Сделайте это привычным способом при помощи параметров `sec` или `#num` и необязательного сдвиг\_времени по аналогии с функциями **avg**, **count**, **delta**, **max**, **min** и **sum**.

#### 13.9.2.2 Горизонт предсказания

(Только **forecast**)

Параметр `time` определяет, насколько далеко в будущее Zabbix будет экстраполировать зависимости, которые ему удастся найти в исторических данных. Независимо о того, используете ли вы сдвиг\_времени или нет, время всегда отсчитывается от текущего момента.

### 13.9.2.3 Достигаемый порог

(Только **timeleft**)

Параметр порог определяет значение, которого должен достичь анализируемый элемент данных, нет никакой разницы, сверху или снизу. После того, как мы определили  $f(t)$  (смотри ниже), мы должны решить уравнение  $f(t) = \text{порог}$  и вернуть ближайший к текущему моменту корень, который находится справа от текущего момента, или вернуть 99999999999.9999, если таковых корней нет.

Когда значения элемента данных приближаются к порогу, а затем пересекают его, **timeleft** делает вывод, что пересечение уже находится в прошлом, и поэтому переключается на следующее пересечение с уровнем порога, если таковое имеется. Наилучшим решением является использование прогнозирований наряду с обычными диагностиками проблем, а не заменой одним на другое.<sup>1)</sup>

### 13.9.2.4 Функции аппроксимации

По умолчанию аппроксимация является линейной (*linear*) функцией. Если наблюдаемая система более сложная, вы можете выбрать один из следующих вариантов.

аппроксимация	$x = f(t)$
линейная ( <i>linear</i> )	$x = a + b*t$
полином ( <i>polynomial</i> ) <sup>2)</sup>	$x = a_0 + a_1*t + a_2*t^2 + \dots + a_n*t^n$
экспоненциальная ( <i>exponential</i> )	$x = a*\exp(b*t)$
логарифмическая ( <i>logarithmic</i> )	$x = a + b*\log(t)$
степенная ( <i>power</i> )	$x = a*t^b$

### 13.9.2.5 Режимы

(Только **forecast**)

Каждый раз, когда вычисляется функция триггера, данные запрашиваются из указанного периода истории и по полученным данным строится указанная аппроксимация. Поэтому, если данные немного изменятся, то и построенная аппроксимация немного изменится. Если мы будем просто рассчитывать значение аппроксимирующей функции в заданный момент времени в будущем, то вы ничего не будете знать о том, как согласно прогнозу будет меняться анализируемый элемент данных между текущим моментом и этим моментом в будущем. При некоторых параметрах аппроксимации (вроде *polynomial*) просто лишь одно значение из будущего может ввести в заблуждение.

режим	результат forecast
значение ( <i>value</i> )	$f(\text{сейчас} + \text{время})$
максимум ( <i>max</i> )	$\max_{\text{сейчас} \leq t \leq \text{сейчас} + \text{время}} f(t)$
минимум ( <i>min</i> )	$\min_{\text{сейчас} \leq t \leq \text{сейчас} + \text{время}} f(t)$
дельта ( <i>delta</i> )	$\text{max} - \text{min}$
среднее ( <i>avg</i> )	среднее значение $f(t)$ ( $\text{сейчас} \leq t \leq \text{сейчас} + \text{время}$ ) в соответствии с определением

### 13.9.3 Подробности

Для того, чтобы избежать вычислений с большими числами, мы рассматриваем штамп времени первого значения в указанном периоде плюс 1 наносекунда как новую точку отсчёта времени (текущие штампы времени порядка  $10^9$ , в квадрате уже  $10^{18}$ , а точность дробных значений около  $10^{-16}$ ). 1 нс прибавляется для того, чтобы все значения времени были положительными, поскольку построение логарифмической и степенной аппроксимаций подразумевает вычисление  $\log(t)$ . Этот сдвиг времени не влияет на линейную функцию, полином и экспоненциальную функции (за исключением более легких и более точных вычислений), но изменяет форму логарифмической и степенной функций.

### 13.9.4 Возможные ошибки

Функции возвращают -1 в следующих ситуациях:

- указанных период не содержит данных;
- математическая операция не задана<sup>3)</sup>;
- сложности вычислений (к сожалению, для некоторых наборов входных данных диапазона и точности формата чисел с плавающей точкой двойной точности становится недостаточно)<sup>4)</sup>.

Никаких предупреждений или ошибок не выдаётся, если выбранная аппроксимирующая функция плохо описывает предоставленные данные или просто данных недостаточно для точного прогноза.

### 13.9.5 Примеры и обработка ошибок

Для получения предупреждения о том, что у вашего узла сети скоро закончится свободное дисковое пространство, вы можете использовать следующее выражение триггера:

```
{host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h
```

Однако, может появиться код ошибки -1 и перевести ваш триггер в состояние проблемы. Вообще говоря, это не плохо, потому что вы получите предупреждение о том, что ваши прогнозирование не работает должным образом, и вам стоит обратить на них внимание, чтобы разобраться почему. Но иногда это плохо, потому что -1 может просто означать, что за последний час не было получено никаких данных о свободном дисковом пространстве данного узла сети. Если вы получаете много сообщений о ложных тревогах, подумайте об использовании более сложного выражения<sup>5)</sup> триггера:

```
{host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h and  
{host:vfs.fs.size[/,free].timeleft(1h,,0)}<>-1
```

Ситуация немного сложнее с **forecast**. Начнём с того, что -1 может перевести, а может и не перевести триггер в состояние проблемы в зависимости от вашего выражения триггера, которое может быть вроде

```
{host:item.forecast(...)}<...
```

или наподобие

```
{host:item.forecast(...)}>...
```

Более того, -1 может быть вполне корректным результатом прогнозирования, если прогнозируемый элемент данных может принимать отрицательные значения. Но вероятность возникновения такой ситуации в реальных условиях пренебрежительно мала (смотрите, [как](#) работает оператор =). Поэтому добавьте

```
... or {host:item.forecast(...)}=-1
```

или

```
... and {host:item.forecast(...)}<>-1
```

если вы соответственно хотите или, наоборот, не хотите рассматривать -1 как проблему.

## Смотрите также

1. [Прогнозирующие функции триггеров \(pdf\) on zabbix.org \[en\]](#)

1)

Например, простой триггер наподобие

```
{host:item.timeleft(1h,,X)} < 1h
```

может перейти в состояние проблемы, когда элемент данных приближается к X, и неожиданно “самоустраниться”, как только значение X достигнуто. Если проблема выражается в том, что значение элемента данных меньше X, используйте:

```
{host:item.last()} < X or {host:item.timeleft(1h,,X)} < 1h
```

Если проблема заключается в том, что значение элемент данных больше X, используйте:

```
{host:item.last()} > X or {host:item.timeleft(1h,,X)} < 1h
```

2)

Степень многочлена может быть от 1 до 6, при этом функция *polynomial1* равнозначна *linear*. Однако, использовать полиномы высоких степеней следует [с осторожностью](#). Если период вычисления содержит меньше точек, чем требуется для определения коэффициентов полинома, степень полинома понизится (например, запрашивается *polynomial5*, но есть только 4 точки, поэтому для аппроксимации будет использована *polynomial3*).

3)

Например, построение экспоненциальной и степенной функций требует логарифмических вычислений значений элемента данных. Если данные содержат нулевые или отрицательные числа, вы получите сообщение об ошибке, поскольку вычисление логарифма возможно только при положительных значениях.

4)

При аппроксимации *linear*, *exponential*, *logarithmic* и *power* выражений все необходимые вычисления можно написать в явном виде. При *polynomial* можно вычислить только *value* без

дополнительных шагов. Вычисление *avg* включает в себя вычисление первообразной полинома (аналитически). Вычисление *max*, *min* и *delta* включает в себя вычисление производной полинома (аналитически) и поиск его корней (численно). Решение  $f(t) = 0$  требует нахождения корней полинома (численно).

5)

Но в этом случае -1 может восстановить триггер из состояния проблемы. Для полной защищённости используйте:

```
{host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h and ({TRIGGER.VALUE}=0 and {host:vfs.fs.size[/,free].timeleft(1h,,0)}<>-1 or {TRIGGER.VALUE}=1)
```

From:

<https://www.zabbix.com/documentation/3.4/> - **Zabbix Documentation 3.4**

Permanent link:

<https://www.zabbix.com/documentation/3.4/ru/manual/config/triggers/prediction>

Last update: **2018/06/19 02:49**

