概念

聚合有兩個面向，橫向(windowing)與縱向(chunk)。

橫向(windowing)聚合

- 依**時間**或**最後收到的資料數量**聚合
- 指**一個 metric**

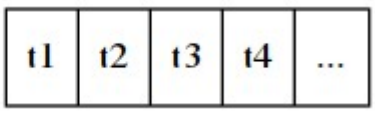
縱向(chunk)聚合

- 依多個 metric 聚合
- 多個 metric 會依時間中樞(time pivot) 規則聚合，如每 20 秒、每一分鐘

橫向聚合(windowing)

見 Windowing 規格

t1 代表最舊的資料，數字越大代表越新的資料



語言定義

- 每一個 window 都以 `$w` 為內建變數名
- 每一個 window 以 `.` 串接多個函式
- 函式
  - window 函式(window function) - 由一個 window 產生另一個 window
  - 化約函式(reduce function) - 由一個 window 產生一個值
    - 結果只能為數字或 null
- 任何運算子有 `null` 值，結果為 `null` 值
  - `null + 20` 等於 `null`

EBNF

```
expr = term
  | expr, "+", term
  | expr, "-", term;

term = factor
  | term, "*", factor
  | term, "/", factor;

factor = primary
  | "-", factor
  | "+", factor;

primary = "(", expr, ")"
  | window reduce
  | number;

window reduce = "$w", [ window functions ], ".", reduce functions;

window functions = { '.', function object };

reduce functions = function object, { '.', function object };
```

```
function object = function name, "(", [ function args ] , ")";

function name = alpha, { letter };

function args = argv, { ",", argv };

argv = number | '"', {string} , '"';

number = ["+" | "-"], digit, {digit}, [ ".", digit, {digit} ];

letter = "_" | alpha | digit ;
alpha = 'a-zA-Z'; (* regexp *)
digit = '0-9'; (* regexp *)

string = '\\' | '\"' | any character;
```

範例

Owl 告警對應計算

- `all(#3)` - `$w.last(3).filter(">=", 30).count()` - 取出最近 3 個值大於 30 的數量
- `max(#3)` - `$w.last(3).max()` - 取出最近 3 個值的最大值
- `min(#3)` - `$w.last(3).min()` - 取出最近 3 個值的最小值
- `sum(#3)` - `$w.last(3).sum()` - 取出最近 3 個值的加總
- `avg(#3)` - `$w.last(3).avg()` - 取出最近 3 個值的平均
- `diff(#3)` - `$w.last(3, 2).map_diff().filter(">", 30).count()` - 取出最後與最近 3 個值的差大於 30 的數量
- `pdiff(#3)` - `$w.last(3, 2).map_pdiff().filter(">", 0.4).count()` - 取出最後與最近 3 個值的差比例大於 40% 的數量

跳動

- `$w.filter(">=", 50).count() / $w.count()` - 取出大於 50 的值與最近 N 個值的跳動比例

window 函式(window functions)

Doc convention:

1. `window[i]` - Every element in the window

Function	Description	Example

`filter(<op>, <expr>)`	filter the elements
	`<op>` - `">="`, `"<="`, `"="`, `">"`, `"<"`, `"!="`
	`<expr>` - could be any `expr`
		`$w.filter(">=", 3.1)` - keep the elements which their value are greater than or equal to *3.1*
		`$w.filter(">", $w.avg())` - keep the elements which their value are greater than average value

`first_of([<number> [, <start>]])`	Extract data from oldest data `<number>` - The maximum number of data to be extracted `<start>` - Start(1st element is `1`) from element	`$w.first_of()` - extract the oldest element `$w.first_of(3)` - extract the oldest 3 elements `$w.first_of(3, 4)` - extract the oldest 4th ~ 6th elements
`last_of([<number> [, <start>]])`	Extract data from newest data `<number>` - The	`$w.last_of()` - extract the newest element `$w.last_of(3)` - extract the newest

Function	Description	Example
	maximum number of data to be extracted `<start>` - Start(last_of element is `1`) from element	3 elements `$w.first_of(4, 2)` - extract the newest 2nd ~ 5th elements
`map_if(<op>, <expr>, <replacing expr>)`	Replace the value of element with `<replacing expr>` if `window[i] <op> <expr>` is true `<op>` - `">="`, `"<="`, `"="`, `">"`, `"<"`, `"!="` `<expr>` - could be any `expr` `<replacing expr>` - could be any `expr`, or `null`	`$w.map_if("=", $w.med(), -1)` - Gets the window with replacing med value to **-1**
`map_abs()`	Gets the window for value of `abs(<element>)`	`$w.map_abs()` - Converts every element to `abs(<element>)`
`map_diff([<minuend>])`	Gets the window for value of `<minuend> - window[i]` `<minuend>` - default value is `$w.last_of()`	`$w.last_of(5, 2).map_diff().filter(">", 30).count()` - Gets the number of diff values(between recent 5 values and lastest one) which are greater than *30*
`map_pdiff([<minuend>])`	Gets the window for value of `(<minuend> - window[i]) / window[i]` `<minuend>` - default value is `$w.last_of()`	`$w.last_of(3, 2).map_pdiff().filter("<", 10).count()` - Gets the number of pdiff values(between recent 3 values and lastest one) which are less than *10*

化約函式(reduce function)

每一個化約函式會忽略 `null` 值

以下函式只能接在 window function 後面

Function	Description	Example
`avg()`	Average of elements `null` - if there is no **`non-null`** element in window	`$w.avg()` - Gets the average value of elements
`count()`	Counting of elements `0` - if there is no **`non-null`** element in window	`$w.count()` - Gets the counting of elements
`first([index])`	Get the value from lastest elemeunt at `window[index]` `null` - if there is no **`non-null`** element in window	`$w.first()` - Gets the value of lastest element `$w.first(2)` - Gets the value of penultimate element from lastest one
`last([index])`	Get the value from lastest elemeunt at `window[index]` `null` - if there is no **`non-null`** element in window	`$w.last()` - Gets the value of oldest element `$w.last(2)` - Gets the value of penultimate element from oldest one

Function	Description	Example
`max()`	Maximum value of elements `null` - if there is no **`non-null`** element in window	`$w.max()` - Gets the maximum value of elements
`median()`	Median value of elements `null` - if there is no **`non-null`** element in window	`$w.median()` - Gets the median value of elements
`min()`	Minimum value of elements `null` - if there is no **`non-null`** element in window	`$w.min()` - Gets the minimum value of elements
`stdev()`	Standard deviation of elements `null` - if there is no **`non-null`** element in window	`$w.stdev()` - Gets the standard deviation of elements
`sum()`	Sum of elements `null` - if there is no **`non-null`** element in window	`$w.sum()` - Gets the sum of elements

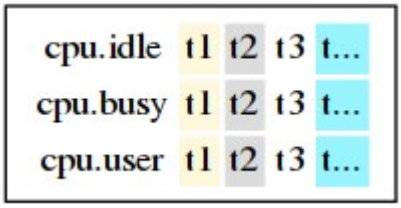
以下函式只能接在 reduce function 後面

Function	Description	Example
`abs()`	Absolute of value `null` - if the value is `null` in window	`$w.map_diff().max().abs()` - Gets the maximum absolute value of diff values on all of the elements
`if_null(<number value>)`	Replace the `null` value to `<number value>` `<number value>` - the value to replace the `null` value	`$w.sum().if_null(30)` - Gets **30** if the value of `sum()` is `null`

縱向聚合(chunk)

依時間中樞(time pivot) 規則聚合，每一個在中樞內的資料，看成一個 chunk

不同 metrics



語言定義

- 每一個 chunk 都以 `$c[<var_name>]` 為存取變數方式
- 每一個 chunk 以 `.` 串接多個函式
- 函式
  - chunk 函式(chunk function) - 由一個 chunk 產生另一個 chunk
  - 化約函式(reduce function) - 由一個 chunk 產生一個值
    - 結果只能為數字或 null
- 任何運算子有 `null` 值，結果為 `null` 值
  - `null + 20` 等於 `null`
- 支援的聚合環境
  - 以 `$ctx` 存取

EBNF

```
expr = term
  | expr, "+", term
  | expr, "-", term;

term = factor
  | term, "*", factor
  | term, "/", factor;

factor = primary
  | "-", factor
  | "+", factor;

primary = "(", expr, ")"
  | chunk reduce
  | number;

chunk reduce = "$c", "[", var name , "]", [ chunk functions ], ".", reduce
functions;

var name = '"', alpha, { letter with dot } , '"';

chunk functions = { '.', function object };

reduce functions = function object, { '.', function object };

function object = function name, "(", [ function args ] , ")";

function name = alpha, { letter };

function args = argv, { ",", argv };

argv = number | '"', {string} , '"';

number = ["+" | "-"], digit, {digit}, [ ".", digit, {digit} ];

letter with dot = "." | letter;
letter = "_" | alpha | digit;
alpha = 'a-zA-Z'; (* regexp *)
digit = '0-9'; (* regexp *)

string = '\\' | '\"' | any character;
```

範例

- 有上報的機器中，"cpu.idle" 大於 60% 的比例: `$c[cpu.idle].filter(">=", 0.6).count() / $ctx.num_effective_endpoints()`
- 符合機器數量減去上報的機器數量: `$ctx.num_match_endpoints() - $ctx.number_effective_endpoints()`

聚合環境函式

以 `$ctx` 存取

Function	Description	Example
`$ctx.num_effective_endpoints()`	Number of endpoints(by effective, reported metrics)	`$ctx.num_effective_endpoints()` - Gets the number of endpoints, which have reported metrics
`$ctx.num_match_endpoints()`	Number of endpoints(matching filter)	`$ctx.num_match_endpoints()` - Gets the number of endpoints, which match the filter of endpoints

Function	Description	Example
`$ctx.num_var()`	Number of defined variables	`$ctx.num_var()` - Gets the number of defined variables
`$ctx.num_null_var()`	Number of null variables	`$ctx.num_null_var()` - Gets the number of null variables

Chunk 函式

Doc convention:

1. `chunk[i]` - Every element in the chunk

Function	Description	Example

`filter(<op>, <expr>)`	filter the elements
	`<op>` - `">="`, `"<="`, `"="`, `">"`, `"<"`, `"!="`
	`<expr>` - could be any `expr`
		`$c[xx.oo].filter(">=", 3.1)` - keep the elements which their value are greater than or equal to *3.1*
		`$c[xx.oo].filter(">", $c[xx.oo].avg())` - keep the elements which their value are greater than average value

`map_if(<op>, <expr>, <replacing expr>)`	Replace the value of element with `<replacing expr>` if `chunk[i] <op> <expr>` is true
	`<op>` - `">="`, `"<="`, `"="`, `">"`, `"<"`, `"!="`
	`<expr>` - could be any `expr`
	`<replacing expr>` - could be any `expr`, or `null`
		`$c[xx.oo].map_if("=", $c[xx.oo].med(), -1)` - Gets the chunk with replacing med value to **-1**

`map_abs()`	Gets the chunk for value of `abs(<element>)`	`$c[xx.oo].map_abs()` - Converts every element to `abs(<element>)`

化約函式(reduce function)

每一個化約函式會忽略 `null` 值

以下函式只能接在 window function 後面

Function	Description	Example
`avg()`	Average of elements `null` - if there is no **`non-null`** element in window	`$c[xx.oo].avg()` - Gets the average value of elements
`count()`	Counting of elements `0` - if there is no **`non-null`** element in window	`$c[xx.oo].count()` - Gets the counting of elements
`max()`	Maximum value of elements `null` - if there is no **`non-null`** element in window	`$c[xx.oo].max()` - Gets the maximum value of elements
`median()`	Median value of elements `null` - if there is no **`non-null`** element in window	`$c[xx.oo].median()` - Gets the median value of elements
`min()`	Minimum value of elements `null` - if there is no **`non-null`** element in window	`$c[xx.oo].min()` - Gets the minimum value of elements
`stdev()`	Standard deviation of elements `null` - if there is no **`non-null`** element in window	`$c[xx.oo].stdev()` - Gets the standard deviation of elements

Function	Description	Example
`sum()`	Sum of elements `null` - if there is no **`non-null`** element in window	`$c[xx.oo].sum()` - Gets the sum of elements

以下函式只能接在 reduce function 後面

Function	Description	Example
`abs()`	Absolute of value `null` - if the value is `null` in window	`$c[net.output].map_diff().max().abs()` - Gets the maximum absolute value of diff values on all of the elements
`if_null(<number value>)`	Replace the `null` value to `<number value>` `<number value>` - the value to replace the `null` value	`$c[net.output].if_null(30)` - Gets **30** if the value of `sum()` is `null`

時間中樞定義

若一個時間中樞取得 **0 筆或超過一筆**以上的資料時，可用橫向聚合(windowing)的語法來定義選擇的資料:

- `$w.last()` - 選擇最新的一筆資料(大於等於二筆資料時)
- `$w.last().if_null(-1)` -
    1. 選擇最新的一筆資料(大於等於二筆資料時)
    2. 以 **-1** 為預設值(沒有對應的資料時)
- `$w.avg()` - 以平均值為資料(或 null 值若為 0 筆)

Last modified on 2017-07-24T13:58:08+08:00