Placement Constraint Expression Syntax Specification

A *single* placement constraint expression contains multiple tokens separated by comma, each one is either a *predefined token*, or a value for a *predefined property*, all tokens must be set in certain order according to the syntax. Generally the syntax for a single placement constraint expression looks like:

$$PC = Token_1 , \ldots , Token_n$$

Predefined tokens include:

- In : affinity to targets
- Notin: anti-affinity to targets
- Node: evaluate the constraint in node scope
- Rack: evaluate the constraint in rack scope

Value for a *predefined property* includes:

- Value for *Allocation Tag*
  - Each tag is a string less than 256 char which contains no space or comma
- Value for *Node Attribute*
  - Use format “key=value” or a “key=[v1:v2:v3]” when attach multiple values to a given attribute. Each value is a string less than 256 char which contains no space or comma.

Multiple expressions can be joined to a single conjunction constraint expression with colon, like following:

$$PC = PC_1 : PC_2 : \ldots : PC_n$$

Such expression means PC is satisfied only when all child PCs are all satisfied. Further, multiple constraint expressions can be joined to a single conjunction constraint expression with an operator AND or OR, such as

$$PC_{and} = AND(PC_1 : PC_2 : \ldots : PC_n)$$
$$PC_{or} = OR(PC_1 : PC_2 : \ldots : PC_n)$$

PC_{and} is satisfied only when all child PCs are all satisfied. PC_{or} is satisfied as long as any child PC is satisfied. Each child PC can by any form of single constraint expression or conjunction constraint expression, so they can be nested in all sorts, such as:

```
AND(PC1:AND(PC2:PC3))
```

See more in next section about single placement constraint expression format.

Single Placement Constraint Expression

Anti-affinity/affinity constraints

Usage

```
[in | notin], [node | rack], target_allocation_tag, …
[in | notin], [node | rack], node_attribute=[attribute_value:...]
```

Define a placement constraint that restricts the placement to be affinity or anti-affinity with one or more allocation tags or node attributes.

Examples

(1) `notin,node,foo`
Do not place me on any node that has foo tag

(2) `in,rack,foo`
Do place me on a node from a rack that has foo tag

(3) `notin,node,host=h1234`
Do not place me on a node whose host attribute value is h1234

Cardinality constraints

Usage

```
cardinality, [node| rack], target_allocation_tag, …,
[min_cardinality], [max_cardinality]
```

Examples

(1) `cardinality,node,foo,0,1`
Do place me on a node where before the allocation happens, this node has at least 0 foo tag, and at most 1 foo tags. So a node is applicable for this allocation if this node has 0 or 1 foo tag before the allocation.

Tips

Anti-affinity and affinity constraint expression can be expressed by cardinality constraint, if (min, max) is (0, 0), this equals to anti-affinity constraint; if it is (0, Int.MAX), this equals to affinity constraint.

Conjunction constraints

Usage

`[AND | OR](PC1: … : PCn)`

Examples

(1) `AND(IN,NODE,foo:cardinality,node,bar,0,1)`

Do place me on a node that has foo tag and at the same time it cannot has more than 1 bar tag.

(2) `OR(IN,NODE,hbase-m:IN,NODE,hbase-rs)`

Do place me on a node which has hbase-m tag or a node which has hbase-rs tag.

Nested Forms

Examples

(1) `AND(IN,NODE,foo:OR(IN,NODE,bar:IN,NODE,moo))`

Do place me on a node which has foo and bar tag, or foo and moo tag.