Merge Constraints

Convention

- Global constraint (**GC**), App-constraint (**AC**), Request-constraint (**RC**)
- If constraint A is more restrictive than B, we say **A > B**

Solution

We have 3 levels of constraints, i.e GC, AC and RC. When check if a given scheduling request can be placed on a node, we do a merge on them with CC=AND(GC, AC, RC) and return us a composite constraint. Subsequently we check if CC could be satisfied.

This ensures us that every level of constraint is satisfied. This means, while validating a placement constraint, we only validate the constraint itself (see more in YARN-6621), we don't validate if a constraint is valid with respect to its upper level constraints. This is due to the difficulty to compare two constraints, as they may have different scope, multiple expressions, different target keys.

User might be confused that why the actual allocation is not expected like what the placement constraint they set in their requests, because they might be restricted by upper level constraints. We need a way to dump all constraints that affected to an unsuccessful placement.

Examples

Take AC, GC as examples, as it is same with AC, RC etc.

1. AC < GC or AC > GC

i) GC: max-cardinality 3, node
AC: max-cardinality 5, node

ii) GC: max-cardinality 5, node
AC: max-cardinality 3, node

Both cases, AC registration succeed as we don't valid if AC is more restrictive than GC or not. But when this application starts to submit requests, both GC and AC will be enforced so on each node, there is maximumly 3 containers.

2. AC and GC has no overlapping

GC: anti-affinity with tag(“foo”), node
AC: max-cardinality 3, node

There is no overlapping between GC and AC, so both needs to be checked.

3. AC and GC are partially overlapping

GC: anti-affinity with tag(“foo”) on node
AC: anti-affinity with tag(“foo”, “bar”) on node

AC is actually more restrictive than GC as it has one more exclusive tag “bar”. AND(GC, AC) checks both constraints but actually only AC is required. (This introduces a bit overhead but we can further improve this by remove duplicate constraints in the evaluation code)