SCHEDULING

Scheduling

Filtering (Predicates)

The scheduler goes through a set of filters, or predicates, to find available nodes, then ranks each node using priority functions. The node with the highest rank is selected to run the Pod.

```
predicatesOrdering = []string{CheckNodeConditionPred, GeneralPred,
HostNamePred, PodFitsHostPortsPred,MatchNodeSelectorPred, PodFitsResourcesPred,
NoDiskConflictPred,PodToleratesNodeTaintsPred,
PodToleratesNodeNoExecuteTaintsPred,CheckNodeLabelPresencePred,
checkServiceAffinityPred,MaxEBSVolumeCountPred,
MaxGCEPDVolumeCountPred,MaxAzureDiskVolumeCountPred,
CheckVolumeBindingPred,NoVolumeZoneConflictPred,
CheckNodeMemoryPressurePred,CheckNodeDiskPressurePred,
MatchInterPodAffinityPred}
```

The predicates, such as **`PodFitsHost`** or **`NoDiskConflict`**, are evaluated in a particular and configurable order. In this way, a node has the least amount of checks for new Pod deployment, which can be useful to exclude a node from unnecessary checks if the node is not in the proper condition.

For example, there is a filter called **`HostNamePred`**, which is also known as **`HostName`**, which filters out nodes that do not match the node name specified in the pod specification. Another predicate is **`PodFitsResources`** to make sure that the available CPU and memory can fit the resources required by the Pod.

The scheduler can be updated by passing a configuration of **`kind: Policy`**, which can order predicates, give special weights to priorities, and even **`hardPodAffinitySymmetricWeight`**, which deploys Pods such that if we set Pod A to run with Pod B, then Pod B should automatically be run with Pod A.