kubernetes #70898 issue : #70622

commit b4fd11512ac3cce6e7932a08db77db798167af1b

问题：被preempting的pod没有按照预期的方式preempted，它们可能会先于nominate pod被分配到对应的nominate node上，反复被preempt，直到nominate pod被分配到nominate node上。这会导致scheduler性能下降。

原因：在原来的scheduler实现中，一旦从队列中pop一个pod，scheduler就会从队列的nominate记录中删除掉该pod。原来的nominated pod已经取消了自己的nominate标记，被preempt的pod可能会再一次被scheduler分配，占据了原来的nominate pod应有的位置。

refs.
The most significant change introduced in this PR is: when popping a pod, scheduler doesn't update internal cache from nominatedMap immediately. Instead, it invalidates the cache until the pod is bound.

Why this? It's because in a very rare case: when a high priority pod comes in, and it's unschedulable (failed in scheduling) (1), it got a chance to try "preemption" (2) and preempt low priority pods (3) to make room.

During phase (1), in function Error() (1.1), it's put back into unschedulableQ where cache nominatedMap is being re-updated, the key point here is: the function is asynchronous (in a goroutine). In other words, after (3) is finished, a backfill pod for the preempted low priority pod (suppose it's managed by a deployment/replicaset) can be spawned and come into scheduling cycle, and it happens prior to (1.1). At this moment, it doesn't know a Nominated pod has been there (as cache hasn't been re-updated), then it's created and enters running state, but it will definitely be preempted again. So this case can happen again and again, although not endless, but really wastes resources to do unnecessary scheduling/preemption.

复现：#70622

修复：

```
@@ -433,6 +433,10 @@ func (sched *Scheduler) assume(assumed *v1.Pod,
host string) error {
                })
                return err
        }
+       // if "assumed" is a nominated pod, we should remove it from
internal cache
+       if sched.config.SchedulingQueue != nil {
+
sched.config.SchedulingQueue.DeleteNominatedPodIfExists(assumed)
```

```
+       }

        // Optimistically assume that the binding will succeed, so we
need to invalidate affected
        // predicates in equivalence cache.
```